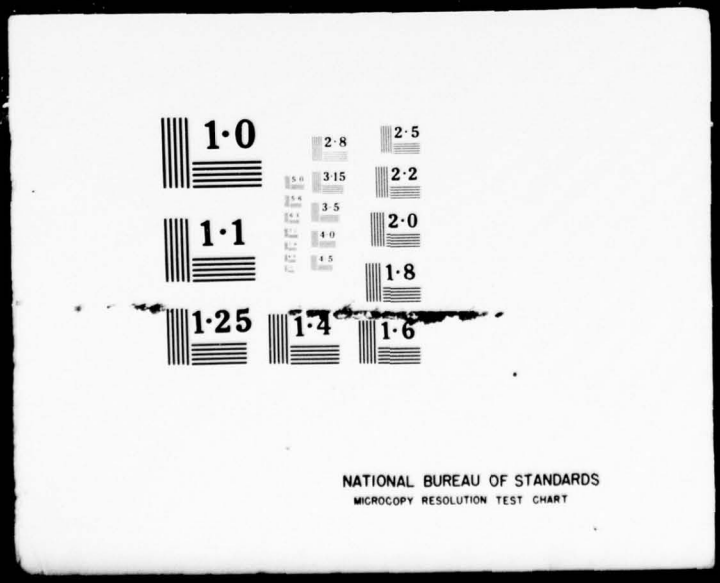
AD-A048 156 WHITE SANDS MISSILE RANGE N MEX INSTRUMENTATION DIRE--ETC F/G 17/8
SEGMENTATION AND STRUCTURE ANALYSIS FOR REAL-TIME VIDEO TARGET --ETC(U)
OCT 77 K FUKUNAGA , A L GILBERT, M K GILES

UNCLASSIFIED STEWS-ID-77-1 NL

1 OF 3
AD A048156



TECHNICAL REPORT

STEWS - ID - 77-1

SEGMENTATION AND STRUCTURE ANALYSIS FOR REAL-TIME VIDEO TARGET TRACKING

OCTOBER 1977

FINAL REPORT

Approved for public release; distribution unlimited

INSTRUMENTATION DIRECTORATE
US ARMY WHITE SANDS MISSILE RANGE
WHITE SANDS MISSILE RANGE, NEW MEXICO 88002

Destroy this report when no longer needed. Do not return it to the originator.

Disclaimer

The findings of this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: STEWS-ID-77-1
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle): SEGMENTATION AND STRUCTURE ANALYSIS FOR REAL-TIME VIDEO TARGET TRACKING.

5. TYPE OF REPORT & PERIOD COVERED: Final Report, May-Aug 77

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s): K. Fukunaga, A. L. Gilbert, M. K. Giles, O. R. Mitchell, R. D. Short, and J. M. Taylor

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Commander
US Army White Sands Missile Range
ATTN: STEWS-ID-T
White Sands Missile Range, New Mexico 88002

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
DA Project No.
1T161101A91A
Work Unit No.: DA 0M1442

11. CONTROLLING OFFICE NAME AND ADDRESS
Commander
US Army White Sands Missile Range
ATTN: STEWS-ID
White Sands Missile Range, New Mexico 88002

12. REPORT DATE: Oct 77

13. NUMBER OF PAGES: ~~285~~ 285

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS. (of this report): Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

DDC RECEIVED JAN 6 1978 F

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

1T161101A91A

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Image Processing
Real Time
Video Tracker
Digitized Images
Image Segmentation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Processing techniques for image segmentation and structure analysis were investigated at White Sands Missile Range (WSMR) during the period May through August 1977. An image processing facility was established at WSMR and used to develop and evaluate image processing algorithms for implementation in future real-time video tracker/processor systems. A simulation of a Real-Time Video Tracker, developed under contract by New Mexico State University, was also implemented at WSMR's image processing laboratory, and a library of 20 digitized tracking

over

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

184230

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Block 20. ABSTRACT (Cont'd)

images was created. Additional tracking imagery, including sequential video frames, is now being added to the digitized image library which will be used to test and evaluate real-time processing algorithms.

Detailed descriptions of the systems hardware and software used in the image processing laboratory are presented along with the segmentation and structure analysis algorithms and the results obtained by processing a few of the digitized images.

PREFACE

This report documents work performed during the period May through August 1977. Major accomplishments include the development and implementation of an image processing laboratory at White Sands Missile Range (WSMR), the development of extensive image processing and system software, the creation of a library of twenty digitized video frames containing typical WSMR target imagery, and the implementation of the New Mexico State University (NMSU) Real-Time Video Tracker Simulation at WSMR. Principal investigators included Dr. K. Fukunaga, Dr. O. R. Mitchell, and Mr. R. D. Short of Purdue University; Dr. J. M. Taylor of NMSU; and Dr. A. L. Gilbert and Dr. M. K. Giles of WSMR.

Expert assistance in programming was provided by Dr. R. Machuca of NMSU and Mr. D. McGinn, University of Tulsa. Mr. Yee Hsun U of NMSU assisted in the tracker simulation. Final software correction, implementation and documentation were accomplished with the help of Mr. J. Hynes and Mr. A. Garcia of WSMR. Mr. R. Ortiz of WSMR prepared the final report manuscript with help from Mr. M. Ramos and Mrs. R. Granger.

Portions of the text were written by each of the principal investigators. Dr. M. K. Giles compiled and edited the final report.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Cont'd)

LIST OF ILLUSTRATIONS (Cont'd)

INTRODUCTION

White Sands Missile Range (WSMR), in cooperation with New Mexico State University (NMSU), is developing a Real-Time Video Tracking System which will incorporate enough real-time "intelligence" to overcome the problems commonly encountered in contrast and correlation television trackers. Contrast trackers are easily fooled by a cluttered scene while correlation trackers require too much computation for reasonable picture sizes and are not easily adapted to significant scene changes. Changing aspect angles, missile staging, multiple targets, and obscurations can defeat both contrast and correlation trackers. The Real-Time Video Tracking System uses high-speed microprocessors together with carefully developed image processing models to extract, recognize, and track the target under many types of adverse conditions.

A team of research personnel from Purdue University was employed at WSMR during the summer of 1977 in order to complement the work being done on the WSMR/NMSU Real-Time Video Tracking System and to help establish an image processing facility at WSMR. The Purdue team generated new ideas for real-time image segmentation and structure analysis which will complement and enlarge the capabilities of the WSMR/NMSU system. The WSMR/NMSU Real-Time Video Tracking System is described in detail in the final report for Contract DAAD07-76-C-0024, "A Preprototype Real-Time Video Tracking System," 17 January 1977.

NMSU personnel contributed to the summer effort by providing valuable technical support to the Purdue and WSMR research personnel and by successfully implementing the NMSU Real-Time Video Tracker Simulation Program on the PDP 11/35-Tektronix 4014 system at WSMR.

This report contains the following sections:

SECTION 1. IMAGE PROCESSING SYSTEM DEVELOPMENT AT WSMR

The ability to digitize video, store data on floppy disks, process images using various techniques, display processed gray level pictures, and write digital images on magnetic tape was developed for WSMR. This section discusses the computer interfacing to peripherals and system programs developed.

SECTION 2. SEGMENTATION AND STRUCTURE ANALYSIS FOR REAL-TIME TARGET TRACKING

This section reviews many of the theoretical concepts developed this summer. Information in this section forms the basis for follow-on research proposals.

SECTION 3. PREPROCESSING AND SEGMENTATION ON THE PDP-11

This section discusses techniques of preprocessing and displays results using sample tracking data.

SECTION 4. SEGMENTATION AND STRUCTURE ANALYSIS ON THE UNIVAC 1108

This section discusses programs developed for simulating shape distortion, extracting contour information, analyzing target characteristics, and performing data transfer operations on the UNIVAC 1108.

SECTION 5. STRUCTURAL ANALYSIS OF NOISY IMAGES USING ONE DIMENSIONAL CONTOUR DESCRIPTORS

This section is a detailed technical paper which develops the pulse sequence image contour representation, the image complexity and similarity measures, and the effects of noise on this structural analysis technique.

SECTION 6. RELATIONSHIP OF RESEARCH TO THAT OF NMSU

SECTION 7. REAL-TIME VIDEO TRACKER SIMULATION

This section discusses the major contributions of NMSU to the summer effort. A description of the Real-Time Video Tracker Program is included.

SECTION 8. COMPUTER PROGRAMS

SECTION 9. SAMPLE TRACKING IMAGERY

SECTION 1

IMAGE PROCESSING SYSTEM DEVELOPMENT AT WSMR

A new PDP 11/35 with two floppy disk drivers and a Tektronix 4014 graphics terminal were acquired by the Advanced Technology Office, Instrumentation Directorate (ID-T) this summer. Also available in the building was a PDP 11/55 and a PDP LSI-11, all with compatible disk drivers.

DIGITIZER

A custom-built digitizer arrived from ISI, Inc., and was debugged, interfaced, and used this summer for converting video frames into two 512 x 240, 8-bit, pixel fields. The present configuration for use of the system is shown in Figure 1.1.

The video tape is played and when an interesting tracking situation occurs, the video disk record button is pushed. A single frame is stored on the disk and either field can be digitized by the computer. A description of the interface and software for the digitizer can be found at the end of this section, and actual programs for the digitizer are given in Section 8. Sample imagery is in Section 9.

It seems feasible to replace the video train up to the digitizer with a slide projector, screen, and TV camera. This would allow digitization of sequential tracking frames of video made through a cinetheodolite system. The video disk is now being refurbished in order to allow disk storage and subsequent digitization of sequential frames of video from a video tape or directly from a television camera.

MAGNETIC TAPE OUTPUT

A Bright magnetic tape recorder with write-only electronics was available and was interfaced to the computer. This allows the distribution of digitized tracking imagery to other computers and research installations. A description of the interface and programs can be found at the end of this section. Actual programs for the magnetic tape output are given in Section 8.

SYSTEMS PROGRAMS

The following system programs have been written; the actual code is given in Section 8 and includes some documentation.

PRINTRONIX LINE PRINTER IMAGE DISPLAY PROGRAMS

LPORIG

Displays gray level output on the Printronix line printer connected to the PDP/LSI-11. This will directly display raw data written by any of the digitizer programs. It expects data stored by 240 pixel columns. It uses a 3 x 3 dot pattern and can create 10 gray levels. It requests the output file, octal black level, and octal increment between display levels.

LPSMAL

Similar to LPORIG but displays transposed pictures assuming 256 pixel rows. It uses a 3 x 7 dot pattern (22 gray levels) so that the aspect ratio is correct for one-field video.

LPLARG

Similar to LPSMAL but assumes 512 pixel rows and prints a left and right page on the line printer.

TEKTRONIX GRAPHICS AND DISPLAY PROGRAMS

HISTP

Computes and plots the histogram of gray levels contained in an entire picture file. The number of columns in the picture and the number of gray levels to be included in each histogram bin are specified by the user. Since each pixel is quantized in 8 bits, 256 gray levels are possible. Hence, if a user specifies one gray level per histogram bin, the program will count the number of pixels present in the picture at each of the 256 gray levels and will plot the resulting 256 numbers as bars on a gray level histogram.

WRP256

Writes a gray level plot to a specified output device (usually the Tektronix display terminal). The input picture file to be displayed must be stored by rows containing 256 pixels per row. Any portion of the input picture file may be displayed at any selected location within the

limits of the display screen. The display is composed of resolution elements written with a 4 x 4 dot matrix and therefore displays 17 half tone shades of gray. The black and white levels of the displayed picture are selected by the user.

WRP512

Similar to WRP256, but requires an input picture file stored in sequential rows with 512 pixels per row. The program compresses each row into 256 pixels before it is displayed by taking the row elements two at a time, computing the average intensity of each pair of elements, and then placing the resulting average intensities in a 256 element row in the compressed picture matrix.

DHIST

Computes and plots the gray level histogram of a specified portion of an input picture file. The number of gray levels to be included in each histogram bin is specified by the user. A half tone gray level plot of the picture is also displayed with user selected dimensions up to 100 x 100 pixels for either 256 or 512 column picture files. The gray level values for black and white in the displayed half tone picture are also selected by the user. In addition, DHIST will display complete picture files as large as 64 x 64 pixels if the trimmed picture file dimensions are 2^n x 2^m with n and m having values from 1 to 6. In this case, the exact picture file dimensions must be entered as the dimensions of the picture to be displayed.

DATA MANIPULATION PROGRAMS

TR256

Transposes a 256 by 240 array of bytes (pixels) stored in sequential 240-element columns to a 240 by 256 array stored in sequential 256-element rows. Digitized picture files obtained from the digitizer programs must be transposed before any other operations can be performed on the data arrays. The only exception is the program LPORIG which displays original digitizer data on the Printronix line printer.

TR512

Similar to TR256, but requires a 512 by 240 input array which it transposes to a 240 by 512 array of bytes stored in sequential 512-element rows.

TRIM

Accepts picture files of any size up to 240 by 512, selects any specified portion of the input picture, and outputs the trimmed picture to a specified output picture file.

TRIM2

Extracts 120 rows and 256 columns of a specified 240 by 512 picture file and writes out the trimmed picture data to a specified output file.

DIGITIZER PROGRAMS

PURPOSE

To store digitized pictures on a floppy disk.

HARDWARE REQUIRED

ISI digitizer, video disk or other "fixed" video input, DR11-C computer interface, floppy disk in DX1 with room on it for picture.

DISK STORAGE REQUIRED

256 by 240 pictures - 120 blocks (1/4 disk); 512 x 240 pictures - 240 blocks (1/2 disk). Output is written on DX1:ORIG.DAT.

VERSIONS

DIG5F1 - 512 x 240 picture from field 1.

DIG5F2 - 512 x 240 picture from field 2.

DIG2F1 - 256 x 240 picture from field 1,
every other column.

DIG2F2 - 256 x 240 picture from field 2.

PROGRAMMING LANGUAGE

MACRO (uses several system MACRO's).

DESCRIPTION

The digitizer is addressed in single point mode so that one point per horizontal line is accessed in real time. Thus, it requires at least 512 x 1/30 ≈ 17 seconds to digitize one 512 x 240 picture. The address of each point to be accessed is loaded into the digitizer, and the vertical address out is watched until it changes, indicating the digitizer is through (or within 2 microseconds of being through) with the present point. A new address is then fed to the digitizer. This process must be repeated fast enough so that the new address arrives before the video signal passes the new addressed point. Presently, the loop is tight enough for this to happen, but any lengthening of the loop might increase processing time by a factor of 240.

Double buffering is used to transfer data from core to the floppy disk so that one buffer is being written to the disk while the other is being filled. Very large buffers are used to take advantage of memory available; however, this makes the SAV version of the program very long which occupies lots of disk space. (The source and object versions are very short.)

OUTPUT FILE DESCRIPTION

Data is output by column with 240 points per column. The last four points in each column are zero because the digitizer hangs sometimes when those last few points are addressed (probably depends on vertical sync width). Each 8-bit pixel is stored in one 8-bit byte on the disk; 0 represents black; and 255, full white. Each disk block holds 512 bytes so that 2-2/15 columns are on each block. The first column is the left side of the TV image, from top to bottom.

WARNING

The output is written on DX1:ORIG.DAT. If such a file already exists, it will be overwritten.

PROGRAM ERRORS

BAD FETCH - Could not find disk handler, probably bad system disk.

BAD ENTER - Could not open output file, probably bad directory on DX1.

WRITE ERROR - Did not write data properly on disk; check to see if disk has contiguous space large enough for the picture.

If program hangs while running, usually the digitizer is not responding with a new address when loaded. Check (1) power on switch on digitizer, (2) cable connection from digitizer to DR11-C, and (3) sync signal on video going to digitizer.

DIGITIZER INTERFACE

The digitizer is cabled to a general purpose parallel interface, DR11-C. The interface has three 16-bit registers: (1) the control and status register (STAT), (2) the output buffer register (ADDR), and (3) the input buffer register (DATA).

THE CONTROL AND STATUS REGISTER (STAT) (767770)

This register is wired so that bit 15 represents conversion complete. This has turned out to be relatively useless since the flag is only down during the time the A/D is actually operating (which is ≈ 2 microseconds which is less than a typical machine instruction time). Bit 7 is the eighth horizontal address output bit from the digitizer. This is not presently used by the programs. Bit 1 is the LOAD GO input (loads vertical address) and is set once before each digitization. Bit 0 is the LOAD HORIZONTAL, and it is set when a new horizontal address is in the address register.

THE OUTPUT BUFFER REGISTER (ADDR) (767772)

This register contains data for the digitizer input. The low nine bits (8 through 0) are the address lines and are used to indicate vertical, horizontal, or mode information. Bit 9 is LDEND and is used for sequential scanning. Bit 10 is LDMODE, and it is used to indicate the digitizer mode (single point, sequential, or interlaced). Bit 15 is the DEX (execute) flag, and it is raised once before each digitization.

THE INPUT BUFFER REGISTER (DATA) (767774)

This register contains data from the digitizer output lines. Bits 7 through 0 represent the digitized value of the addressed point. Bits 12 through 8 are the low-order five bits of the horizontal address. Bits 15 through 13 are the low-order three bits of the vertical address. The address values are not changed until the digitizer video address matches the address loaded; then the output address is changed to match the address loaded. These vertical address bits are presently used as the flag to indicate the digitizer is ready for the next address.

WARNING

The vertical address is unusual. For loading purpose (into ADDR), the high-order bit is the field; i.e., addresses 0 to 353_8 correspond to the first field, and addresses 400_8 to 453_8 correspond to the second field.

For reading purposes (from DATA), the low-order bit is the field; i.e., 000_2 is first row, first field; 001_2 is first row, second field; 010_2 is second row, first field, etc.

The wire lists which follow (Tables 1.1 and 1.2) indicate actual cable connections between the two 40-pin connector plugs on the DR11-C in the computer and the one 50-pin plug on the digitizer.

MAGNETIC TAPE PROGRAMS

PURPOSE

To write gray-level pictures stored on floppy disks onto 7-track magnetic tape.

HARDWARE REQUIRED

Bright tape recorder, DR11-C computer interface, ARL Radar Echo Simulator Box.

VERSIONS

MAG256 - Writes picture files that have 256 points per row, 240 rows.

MAG512 - Writes picture files that have 512 points per row, 240 rows.

PROGRAMMING LANGUAGE

MACRO.

DESCRIPTION

The 7-track magnetic tape accepts six data bits per tape frame. The data must be fed to the recorder as fast as it will accept data or it will write end of record blocks randomly while ignoring some of the data. Thus, a buffer in core must be readied with one record of data. This is then written continuously on the tape. Then a pause occurs while the next record of data is read from the disk to core; and, at the same time, the tape control unit writes an end of record block.

The program initially sets bits in the interface (to stop tape from moving) and then asks for the file on the disk to be transferred to tape. After completion of writing, it requests the next file to be written.

TABLE 1.1. COMPUTER INTERFACE CONNECTIONS

CABLE CONNECTORS
COMPUTER INTERFACE
DR11-C
BERG H856 40-PIN CONNECTORS

CONNECTOR NO. 1		PIN NO. ON DIGITIZER		CONNECTOR NO. 2		PIN NO. ON DIGITIZER	
PIN	NAME	NAME	PIN	PIN	NAME	NAME	PIN
C	OUT00	DIN0	2	H	IN02	DOUT2	21
J	GND	GND	16	J	GND	GND	47
K	OUT01	DIN1	4	K	CSR0	LDHOR	3
L	OUT04	DIN4	10	L	GND	GND	16
M	GND	GND	31	M	IN15	VOUT2	35
N	OUT05	DIN5	12	N	IN14	VOUT1	37
R	OUT06	DIN6	14	P	IN13	VOUT0	39
S	GND	GND	47	R	GND	GND	31
T	OUT07	DIN7	15	S	REQ B	CONV. COMP.	29
U	OUT03	DIN3	8	T	GND	GND	47
V	GND	GND	16	U	IN12	HOUT4	44
W	OUT08	DIN8	13	V	IN11	HOUT3	42
X	OUT09	LDEND	9	W	IN10	HOUT2	40
Y	GND	GND	31	X	GND	GND	16
Z	OUT10	LDMODE	7	Y	IN09	HOUT1	38
CC	GND	GND	47	Z	IN08	HOUT0	36
DD	CSR1	LDGO	11	AA	GND	GND	31
EE	GND	GND	16	BB	IN03	DOUT3	19
JJ	OUT15	DEX	5	CC	IN07	DOUT7	32
KK	GND	GND	31	DD	GND	GND	47
LL	REQA	HOUT8	41	EE	IN06	DOUT6	33
MM	GND	GND	47	HH	IN05	DOUT5	50
NN	OUT02	DIN2	6	JJ	GND	GND	16
PP	GND	GND	16	KK	IN04	DOUT4	17
SS	GND	GND	31	LL	IN01	DOUT1	23
UU	GND	GND	47	MM	GND	GND	31
				PP	GND	GND	47
				SS	GND	GND	16
				TT	IN00	DOUT0	25
				UU	GND	GND	31

TABLE 1.2. DIGITIZER CONNECTIONS

BERG CONNECTOR NO. 1		BERG CONNECTOR NO. 2	
PIN	DIGITIZER PIN	PIN	DIGITIZER PIN
A	OPEN	A	OPEN
B	OPEN	B	OPEN
C	2	C	OPEN
D	OPEN	D	OPEN
E	OPEN	E	OPEN
F	OPEN	F	OPEN
H	OPEN	H	21
J	GND 16	J	GND 47
K	4	K	3
L	10	L	GND 16
M	GND 31	M	35
N	12	N	37
P	OPEN	P	39
R	14	R	GND 31
S	GND 47	S	29
T	15	T	GND 47
U	8	U	44
V	GND 16	V	42
W	13	W	40
X	9	X	GND 16
Y	GND 31	Y	38
Z	7	Z	36
AA	OPEN	AA	GND 31
BB	OPEN	BB	19
CC	GND 47	CC	32
DD	11	DD	GND 47
EE	GND 16	EE	33
FF	OPEN	FF	OPEN
HH	OPEN	HH	50
JJ	5	JJ	GND 16
KK	GND 31	KK	17
LL	41	LL	23
MM	GND 47	MM	GND 31
NN	6	NN	OPEN
PP	GND 16	PP	GND 47
RR	OPEN	RR	OPEN
SS	GND 31	SS	GND 16
TT	OPEN	TT	25
UU	GND 47	UU	GND 31
VV	OPEN	VV	OPEN

OUTPUT FILE DESCRIPTION

The 7-track tape is written at 800 bits per inch (hi density) with odd parity. There is one picture per file and within each file one line per record. The data is packed so that the first frame contains the six most significant bits of the first 8-bit data point. The second frame contains the last two bits plus the first four of the next data point, etc., so that three 8-bits data points are packed into four 6-bit tape frames. The picture data is stored left to right, top to bottom. The tape is written in IBM compatible format.

WARNINGS

1. The first file on the tape is sometimes corrupted. It is safest to write the first file twice.

2. Do not initiate a hardware reset (done by hitting START on the console or typing two consecutive CONTROL C's) while the tape is ONLINE. This will clear the DR11-C register and start the tape moving. The tape must then be rewritten or nothing else added since the tape cannot be rewound to the exact location without read electronics.

3. Be sure the ARL Radar Echo Simulator Box is turned on; the interface uses a 5-volt supply in there.

TAPE RECORDER INTERFACE

The Bright Model 2610 7-track write only tape recorder is cabled to a general purpose parallel interface, DR11-C. The interface has three 16-bit registers: (1) the control and status register (STAT), (2) the output buffer register (ADDR), and (3) the input buffer register (DATA).

THE CONTROL AND STATUS REGISTER (STAT) (767760)

This register is wired so that bit 7 is the tape recorder $\overline{\text{DATA RECEIVED}}$ signal. This read only bit goes to a zero after the tape drive has received the data loaded by the $\overline{\text{DATA READY}}$ line. Bit 1 is the $\overline{\text{DATA READY}}$ line, it should be lowered to a zero whenever new data is in the DATA register and ready to be written on the tape. If the $\overline{\text{DATA READY}}$ line is not lowered when the tape is in position for the next character, the drive will assume an end of record and will proceed to write an end of record block. No hardware connections to the computer can presently tell when this is happening.

THE OUTPUT BUFFER REGISTER (DATA) (767762)

This register holds the data to be written on the tape in the lower order 6-bits (5 through 0). The data should be complemented (zeros and ones interchanged). Bit 15 of this register is $\overline{\text{TAPE GAP}}$ (tape gap not) and will write an end of file (EOF) mark on the tape when lowered to a zero.

THE INPUT BUFFER REGISTER (767764)

This register is not used since the tape drive does not have read capabilities.

WARNINGS

1. Without read capabilities, it is impossible to update a tape. Each tape written must be written from the beginning of the tape.

2. The first file written on each tape is sometimes bad (extra record with parity errors). The safe thing to do is write the first file twice and ignore the first one on the tape.

3. Bit 1 in STAT and bit 15 in DATA must be 1's for the tape to be stopped. The software programs do this, but a hardware INIT pulse (such as caused by rebooting or two consecutive CONTROL C's) will reset the registers to zero, starting the tape to move, and eliminating the possibility of writing anything else on the tape (due to location being lost) unless the entire tape is rewritten.

4. Bit 7 of STAT must have a pull-up register wired to +5 volts on it to operate correctly. This is presently done at the connector in the ARL Radar Echo Simulator Box. This box must be turned on for the +5 volts to exist. If it is not on, the drive will seem OK but the tape written will be unreadable.

The wire list which follows (Table 1.3) indicates actual cable connections between one 40-pin connector on the DR11-C in the computer and the connectors in the ARL Radar Echo Simulator Box.

TABLE 1.3. MAGNETIC TAPE UNIT CONNECTIONS

BERG CONNECTOR			COMMON CONNECTOR		TAPE CONNECTOR		
PIN NO.	PIN LETTER	SIGNAL WAVE	PIN NO.	WIRE COLOR	PLUG NO.	PIN NO.	NAME
3	C	OUT00	5	Green	BJ2	13 Bot	$\overline{1}$
8	J	GND	10	Orange	BJ2	8-13 Top	GND
9	K	OUT01	3	Orange	BJ2	12 Bot	$\overline{2}$
10	L	OUT04	4	Yellow	BJ2	9 Bot	$\overline{A}$
12	N	OUT05	6	Blue	BJ2	8 Bot	$\overline{B}$
15	S	GND	10	Orange	BJ2	8-13 Top	GND
17	U	OUT03	2	Red	BJ2	10 Bot	$\overline{8}$
18	V	GND	10	Orange	BJ2	8-13 Top	GND
26	DD	CSR1	8	Gray	BJ2	4 Bot	Data Ready
27	EE	GND	10	Orange	BJ2	8-13 Top	GND
30	JJ	OUT15	9	Brown	BJ1	6 Top	Tape Mark Gap
31	KK	GND	10	Orange	BJ2	8-13 Top	GND
32	LL	REQA	7*	Purple	AJ2	3 Bot	Data Received
33	MM	GND	10	Orange	BJ2	8-13 Top	GND
34	NN	OUT02	1	Brown	BJ2	11 Bot	$\overline{4}$
35	PP	GND	10	Orange	BJ2	8-13 Top	GND

*Data Received also wired to Pin 2-4-J in ARL Radar Echo Simulator Box. See p. 77 in ARL-TR-75-1.

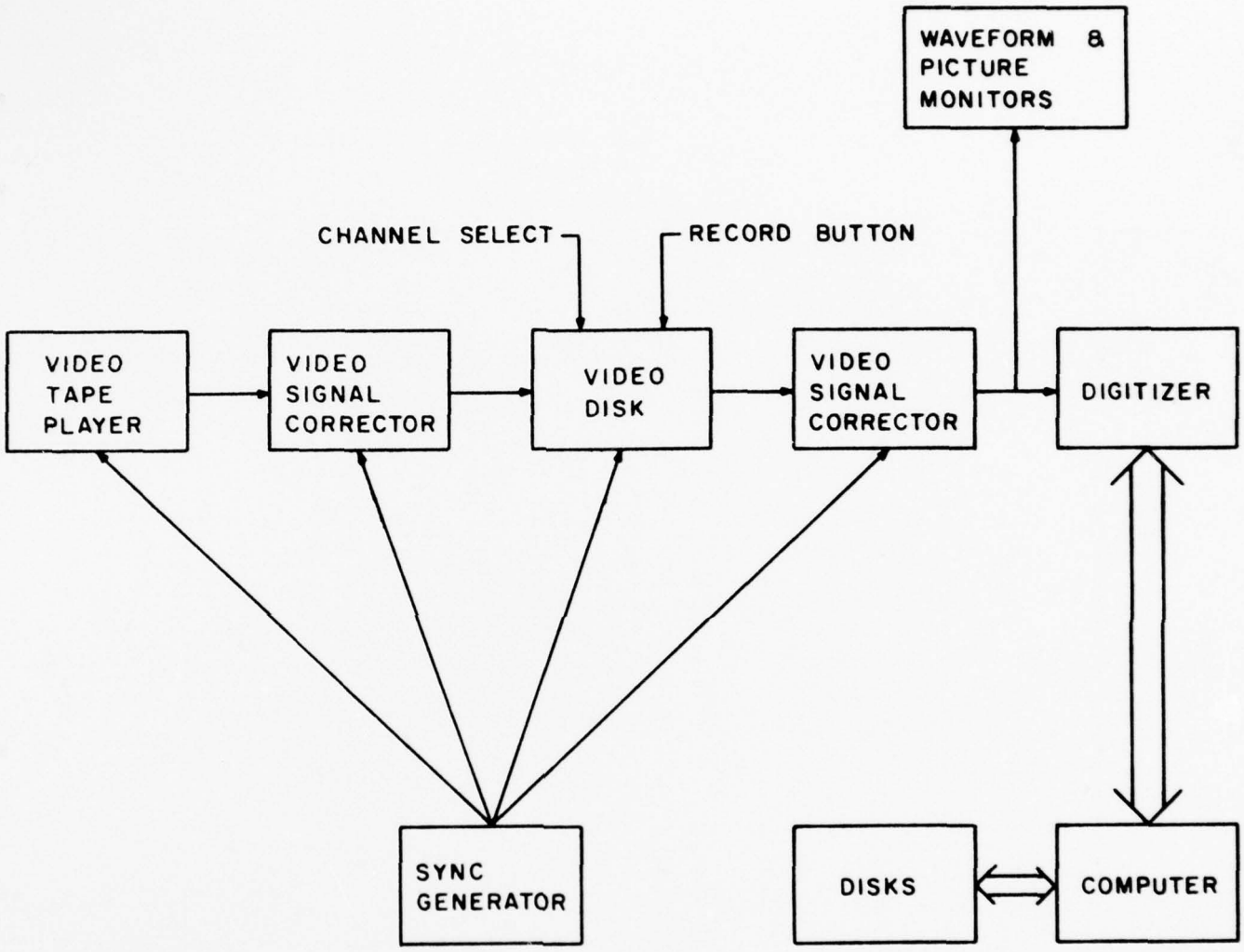


Figure 1.1. Video Digitization Block Diagram

SECTION 2

SEGMENTATION AND STRUCTURE ANALYSIS FOR REAL-TIME TARGET TRACKING

INTRODUCTION

The completed work suggests further effort to design segmentation and structure analysis algorithms that operate reliably on tracking imagery such as that supplied by WSMR. Although real-time implementation is the ultimate objective, and all algorithms will be evaluated with this in mind, the goal of the follow-on effort will be to provide a simulation of the tracker segmentation and structure analysis sections in an off-line mode.

TRACKING SYSTEM REQUIREMENTS

Many applications require the precise location, orientation, and description of moving objects in real time. The work at WSMR this summer has provided exposure to the special requirements of missile and aircraft tracking problems which are best solved by optical imaging tracking systems.

Present optical tracking systems are "one concept" trackers, using either contrast or correlation. The contrast trackers[1] are high speed but lack the versatility to follow a target in complex scenes. The template matching or correlation trackers[2] require a lot of computation for reasonable window sizes and are not easily adapted to changing aspect angles, missile staging, and other common scene changes.

The need for a "smart" optical tracker exists which would operate in real time and could adapt to target orientation and shape changes (due to staging, visible emissions, etc.), and to partial or total obscuration of the target at times.

Presently, NMSU, under a grant from the Army Research Office (DAAG29-76-G-0231), is developing a prototype system with some of the capabilities mentioned. However, the technical aspects of target segmentation and structure analysis need to be studied in much more detail in conjunction with the use of actual video data to develop a system that is robust under the normal variety of tracking situations encountered.

1. R. H. Norris, J. W. Hines, and E. Ashenfelder, "A Digital Video Contrast Tracker," NAECON, 1975.
2. C. W. Winsor, "Area Tracking Using Electro-Optic Signal Processing; TV Area Correlation Tracking," SWIECO, pp. 501-504, 1973.

Purdue University is presently developing image understanding techniques under Grant No. MDA90377G1 from the Defense Advanced Research Projects Agency (DARPA). The present emphasis of the DARPA research is high level information extraction and symbol minipulation using image structure and context. These techniques are applicable to the tracking problem; however, for these methods to fully apply, some lower level techniques must first be developed for preprocessing, target extraction, and target description.

The basic ideas for these processes have been derived over the summer and must now be fully developed and tested using WSMR video data.

DATA

A system has been developed at WSMR this summer to put digitized video data on IBM compatible format 7-track magnetic tape. Twenty digitized video images representing some tracking situation are presently available. Each image is one video field consisting of 512 by 240 points with 256 gray levels. Because the video interlace is not used, the horizontal resolution is twice that of the vertical.

Follow-on research would include digitizing several continuous flights (e.g., 30 frames per second for 2 seconds) and using this data for development and testing of algorithms.

FOLLOW-ON RESEARCH

The research can be divided into two major areas: (1) segmentation of potential targets from the background, and (2) structure analysis of the segmented targets.

Segmentation requires high speed processing of picture points to identify potential target areas and more detailed processing of these areas to extract a binary image of the potential target.

Structure analysis determines if the object extracted is indeed the one to be tracked, the exact location, and orientation of the object, and recent changes that have occurred in the object.

Both the segmentation and structure analysis sections will use information obtained from previous frames that is stored and modified in real time to allow significant changes in the tracked object during a mission.

SEGMENTATION

The segmentation of the target from the background is the most critical timing problem for real-time implementation. At video rates, a new picture element (pixel) arrives approximately every 100 nanoseconds. Thus, either all initial operations must be simple or the area to be processed must be reduced by predicting the target location in the image before processing. It is the opinion of the authors that the system would operate more accurately if the entire picture were processed every frame. This would reduce the severe requirement of accurate prediction of target location in the picture and allow the system to acquire and discriminate interfering objects before they come confusingly close to the target being tracked.

PREPROCESSING

With the large variety of incoming images, it is often necessary to do some preprocessing to make the system more invariant to changes in lighting, viewing angle, and resolution. It is also useful to enhance those aspects of the image which indicate the presence of targets (such as edges and texture of man-made objects). The following preprocessing operations will be investigated: (1) a logarithmic input transformation, (2) a two-dimensional median filter, (3) a two-dimensional human visual system (HVS) filter, and (4) averaging.

These processes are all window operations that can be performed in real time over the entire incoming video picture. The output of these processes will allow the selection of suitable potential target regions for more detailed processing.

Logarithmic Input Transformation

Working with the densities (log intensities) allows the system to be more invariant to illumination and lens opening changes since these produce multiplicative effects in the intensity image and additive effects in the density image. The logarithm also emphasizes the detail in the darker areas of the image which is especially useful to the texture processing algorithms described later. The logarithm is also an approximation to the visual system's action on an incoming pattern[3]. Future research should indicate whether this nonlinear operation is appropriate or necessary for typical video tracking data.

Two-Dimensional Median Filter

The median filter allows for video dropouts including single point or single line mistakes in the analog video chain or the digitizer. A simple median

3. T. N. Cornsweet, Visual Perception, Academic Press, New York, 1970.

filter replaces the center point of each 3 x 3 window with the median of all 9 points in the window. This nonlinear process eliminates high frequency noise but preserves monotonic edges precisely.

Shown in Figure 2.1 is digitized video of an airplane with the contrast enhanced. The result of processing this with a 3 x 3 median filter is shown in Figure 2.2. Notice that edges and objects are preserved but the noise is reduced and the data seems to be more correlated.

Figures 2.3 and 2.4 show the same preprocessing effects on a Cruise missile. The same process is shown in Figures 2.5 and 2.6 on a missile target with a large plume.

Continued research should indicate the advantages and disadvantages of this preprocessing operation for the video tracking data supplied to us.

Two-Dimensional Human Visual System Filter

It is well known that the HVS responds to incoming images with an approximate logarithmic transformation followed by a spatial and temporal bandpass filtering operation. The usefulness of such a spatial bandpass filter in extracting objects for tracking will be investigated.

A suitable filter function which is to be convolved with an input image is shown in Table 2.1. The values are chosen to make computation of the fractional values easy for real-time implementation. Because the video data has twice the resolution in the horizontal direction than in the vertical (only one video field is processed at a time), the HVS filter is modified as shown in Table 2.2. Both filters have a dc response of 0.5 and a gain of 7.5 for a small object which just covers the plus area. The filter shown in Table 2.2 was used to produce Figure 2.7 using Figure 2.6 as the input. Note that edges are emphasized and that the missile (above the plume) is enhanced.

Future research should indicate the appropriateness of such a filter and the parameters best suited to video tracking data.

Averaging Filter

The effects of averaging on images over a small window tend to blur the image slightly, but can be of help in segmenting an image by thresholding. For example, the use of a 5 x 3 averaging window on Figures 2.2 and 2.4 produces the results shown in Figures 2.8 and 2.9, respectively. The general usefulness of this combination median and averaging process will be examined (the median filter removes bad points before they can be smeared by the averager).

TABLE 2.1. TWO-DIMENSIONAL HVS FILTER FUNCTION

0	0	0	0	-.25	0	0	0	0
0	0	0	-.25	-.5	-.25	0	0	0
0	0	-.25	-.25	+.25	-.25	-.25	0	0
0	-.25	-.25	+.5	+1.0	+.5	-.25	-.25	0
-.25	-.5	+.25	+1.0	+1.5	+1.0	+.25	-.5	-.25
0	-.25	-.25	+.5	+1.0	+.5	-.25	-.25	0
0	0	-.25	-.25	+.25	-.25	-.25	0	0
0	0	0	-.25	-.5	-.25	0	0	0
0	0	0	0	-.25	0	0	0	0

TABLE 2.2. MODIFIED HVS FILTER TO ALLOW HALF RESOLUTION IN VERTICAL DIRECTION

0	0	0	-.25	-.75	-.25	0	0	0
0	-.25	-.5	+.25	+1.25	+.25	-.5	-.25	0
-.25	-.5	+.25	+1.0	+1.5	+1.0	+.25	-.5	-.25
0	-.25	-.5	+.25	+1.25	+.25	-.5	-.25	0
0	0	0	-.25	-.75	-.25	0	0	0

REGION GROWING USING WINDOW STATISTICS

After preprocessing the input video data, potential target regions must be extracted. Depending on the data, this task may be a simple thresholding operation or it may be a more complex extraction procedure. Initially, the segmentation will be formulated in terms of measuring statistics over each n x m window in the preprocessed pictures and assigning each window to one of several classes.

The measurements which can be made on an n x m window range from a simple histogram of the data points to a representation in (n)(m)-dimensional vector space. Of course, the simplest representation that will give a proper segmentation is the most desirable. This selection can only be made after experience is gained by segmenting a large amount of typical tracking data.

Once a window parameter is selected, points must be clustered to give appropriate regions. This clustering must be based upon some distance measure between two window parameters. The following window parameters, distance measures, and clustering techniques will be investigated.

First Order Measure

The histogram of the points in each window will be used as the window parameter. The distance measure will be the Bayes error overlap between the two modified histograms:

$$d = 1 - \int \min[f(x)*h_1(x) \ , \ f(x)*h_2(x)]dx \quad , \tag{1}$$

where

$h_1(x)$ and $h_2(x)$ are the gray level histograms over the two windows, and

$f(x)$ is a smoothing and normalizing function to be convolved with each histogram.

The smoothing function is chosen so that the modified histogram has an area of unity and is smoothed so that the distance measured by Equation (1) is not adversely affected by the discrete nature of the original histograms.

An example is shown in Figures 2.10 and 2.11. The x's in Figure 2.10 represent the histogram from window 1 and the 0's represent the histogram from window 2. The functions $p_1(x)$ and $p_2(x)$ in Figure 2.11 represent the smoothed histograms. The shaded region is the integral in Equation (1). This area is equivalent to the probability of error in estimating to which of two density function estimates a sample gray level belongs.

The distance measure is now used to cluster windows that are close together to form segmented regions. Prior frame information should be useful here to provide initial cluster groupings. Windows that are more than a threshold distance away from all known clusters will be used to form new clusters.

Second Order Measure

In tracking situations where first order measures do not allow proper segmentation, a second order measure may be applied. A modified joint histogram relating pairs of points within a window will be assessed:

$$p(x, y) = f(x, y)*h(x, y) \quad , \tag{2}$$

where

$h(x, y)$ is the joint histogram of pairs of adjacent points (co-occurrence matrix), and

$f(x, y)$ is a two-dimensional smoothing and normalizing function.

The co-occurrence matrix has been applied to texture analysis[4] and has shown encouraging results. However, instead of calculating parameters on the matrix as is done in Reference 4, the matrix itself will be used in a two-dimensional version of the overlap error measurement of distance,

$$d = 1 - \iint \min[p_1(x, y) , p_2(x, y)]dxdy \quad . \tag{3}$$

Higher order measures are probably impractical but could be investigated if these initial measurements show obvious shortcomings.

LOCAL EXTREMA INFORMATION

The detection of local gray level extrema in an image has shown to be useful in texture classification and image segmentation[5,6]. The detection algorithm for this operation is as follows.

4. R. M. Hanalick, K. Shanmugan, and I. Dinstein, "Textural Features for Image Classification," IEEE Trans. on Systems, Man, and Cybernetics, SMC-3, pp. 610-621, November 1973.

5. O. R. Mitchell, C. R. Myers, and W. Boyne, "A Max-Min Measure for Image Texture Analysis," IEEE Trans. on Computers, Vol. C-26, pp. 408-414, April 1977.

6. S. G. Carlton and O. R. Mitchell, "Image Segmentation Using Texture and Gray Level," Proceedings of the IEEE Conference on Pattern Recognition and Image Processing, pp. 387-391, June 1977.

In a n x m window, compare the gray level of the center point with those of its two vertical neighbors. If it is above both neighbors, the center point is a local maximum in the vertical direction. If this is the case, compare the center point with each point along each vertical direction until a gray level is encountered which is above the center point's value or until the edge of the window is encountered. The largest differences between gray levels in each vertical direction are then compared and the smallest of the two is retained as the size of the local maximum in the vertical direction. An example is shown in Table 2.3 for a 5 x 7 window.

TABLE 2.3. SAMPLE GRAY LEVELS FOR EXTREMA DETECTION

36	40	47	30	24
33	34	30	32	36
36	40	32	40	30
42	46	45	43	35
36	40	33	47	32
34	42	50	42	40
30	30	20	45	36

The center value is 45. The 47 ends the search in the top direction and the 50 ends the search in the bottom direction. The range is 15 above and 12 below. Therefore, the center point is a local maximum in the vertical direction of size 12. If the point is a local minimum instead of a maximum, the process is down in the same way interchanging the above and below comparison tests.

This process is also done in the horizontal direction. In the example of Table 2.3, the center point is not a local extreme in the horizontal direction. If a point is a local extreme in both horizontal and vertical directions, only the largest of the two is retained at that location. The extrema detection process is equivalent to local maximum and minimum determination following hysteresis smoothing of various amounts.

Figure 2.12 shows the output of such an operation on the image in Figure 2.7. Extrema size is indicated by the gray level. No distinction is made in the figure between maxima and minima or between horizontal or vertical extrema.

Note that the edges emphasized by the HVS filter now are marked by the extrema. The missile orientation can be extracted from its edge information, however, the algorithm must be high level enough so that the missile edge is extracted and not the plume. (The missile orientation angle may not be the same as its flight direction.)

Also, the texture of various regions can be characterized by the types and number of extrema present. The region characterization may be useful for background classification and for plume identification. Parameters for this measurement are extracted by counting the number of various size extrema within a window surrounding each point.

EDGE INFORMATION

The HVS is very sensitive to edges and lines[7]. Patterns containing edges are much more visible than those containing the same signal power but lacking well-defined edges. Also, many objects to be tracked are distinguished from natural environment backgrounds by the presence of edges. It would seem appropriate that one tool that should be included in the tracker repertoire is the ability to detect and use edge information.

Edge information may be included as follows: when edges are passed through the HVS filter and the local extrema detector, they become connected local extrema. A processor will look for connected local extrema and label these extrema to be edge extrema as opposed to texture extrema. These edge extrema can then be used as follows:

1. the presence of an edge will enhance the possibility of a region being listed as a potential target area, and

2. windows which contain edges will be modified so that edge points control the window shape.

Consider Figure 2.13 as an example. In this figure, detected edge points are marked with an "E." To perform a window operation, scanning starts at the center point and moves in the numbered directions, stopping at an edge point or at the window edge. This allows nonrectangular windows along the boundary of targets.

EXTRACTION OF BINARY IMAGES

The purpose of all preprocessing and segmentation techniques described in prior sections is to produce binary images which are potential targets for tracking. Information to be used in the selection of binary images will be:

7. Cornsweet, op. cit.

1. Segmented regions from Region Growing Using Window Statistics Section.

2. Texture information from Local Extrema Information Section.

3. Edge Information Section.

4. Prior frame information.

The initial protocol for potential target selection is shown in the flow chart in Figure 2.14.

STRUCTURE ANALYSIS OF BINARY IMAGES

The purpose of structure analysis in this study is to find structure differences among various targets such as airplanes, helicopters, missiles, and balloons. Here, it is assumed that the target is separated from background by the segmentation processes of the previous sections and all target points are represented as 1's in a binary image. The complexity of a binary image may be used to characterize the structure. This feature and its variations will be simple enough for real-time implementation, yet complex enough to identify the differences of these targets even with various viewing angles.

A FOURIER DESCRIPTOR

A binary image may be characterized by the properties of its contour. Let us express the angle difference of two tangents at S and A in Figure 2.14 as a function of ℓ, the contour length between these two points. An example of the function $\theta(\ell)$ is shown in Figure 2.15. Thus, a two-dimensional image contour is converted into a one-dimensional curve. The Fourier transform of this curve has been called a Fourier descriptor[8,9,10]. The Fourier descriptor has several advantages as an expression of an image. These are

1. invariant under rotation of the image,

2. invariant under translational shift of the image,

8. C. T. Zahn and R. Z. Roskies, "Fourier Descriptors for Plane Closed Curves," IEEE Trans. on Computers, Vol. C-21, pp. 269-281, March 1972.

9. G. H. Granlund, "Fourier Preprocessing for Hand Print Character Recognition," IEEE Trans. on Computers, Vol. C-21, pp. 195-201, February 1972.

10. E. Persoon and K. S. Fu, "Shape Discrimination Using Fourier Descriptors," IEEE Trans. on SMC, pp. 170-179, March 1977.

3. selection of the starting point S appears only as a phase shift, and

4. invariant under scale changes of the image if ℓ is normalized by the entire contour length L, such as $t = 2\pi\ell/L$.

Figure 2.16 shows $\theta(t)$'s for various images. Figure 2.16 also shows an alternative function $d\theta/dt$ which will be used. As is seen in the figures, in $d\theta/dt$, all convex and concave angles appear as negative and positive pulses, respectively. The existence of these pulses has been considered by past researchers as a disadvantage of the Fourier descriptor because of the high frequency components of a pulse. However, as the figures illustrate, this pulse sequence may be used to entirely characterize an image contour.

Before defining the complexity of a binary image and its variations in terms of $d\theta/dt$, let us look at some properties of $d\theta/dt$. Figure 2.17 shows that when an image has a smooth contour, $d\theta/dt$ will be a continuous curve. The locations of the peaks and the areas under the peaks in this figure correspond to the locations and heights of the pulses in Figure 2.16.

Figure 2.18 shows how to treat unclosed line images. These images may be traced twice as if the lines have a finite thickness.

COMPLEXITY MEASURE

Since the function $d\theta/dt$ carries all the information about an image contour, $d\theta/dt$ or its Fourier transform, $F[d\theta/dt]$, may be used for structure analysis of binary images. The advantage of using $d\theta/dt$ is that $d\theta/dt$ is a one-dimensional function rather than two, and all techniques developed for waveform analysis can be utilized. However, it appears that $d\theta/dt$ or $F[d\theta/dt]$ are still too complex for real-time testing, and that there is a need to extract simpler features from it.

The first and simplest feature extracted from $d\theta/dt$ would be the complexity measure defined as

$$C = \int_0^{2\pi} \left|\frac{d\theta}{dt}\right| dt - 2\pi \quad . \tag{4}$$

Noting that $\int d\theta/dt\, dt = -2\pi$ for all closed contours, Equation (4) becomes zero for all convex images. When an image includes concave angles, Equation (4) becomes twice the summation of all concave angles in that image. That is, the proposed complexity measure indicates the number and size of the concave angles. Figure 2.19 shows these concave angles of various target images. Obviously, one cannot expect that a single number can represent many properties of an image. However, measuring concave angles seems to be a reasonable thing to do as the first order characterization of $d\theta/dt$, and

it looks as if C of Equation (4) could be sufficient to classify complex images such as airplanes and helicopters from simple ones such as missiles and balloons.

DESCRIPTION OF BINARY IMAGES

When the complexity measure is not sufficient to classify targets, more information must be extracted from $d\theta/dt$. A comparison of Figures 2.16c and 2.17 shows that the pulse sequence of Figure 2.16c is a simpler representation of the image than the continuous curve of Figure 2.17. Indeed, the pulse sequence representation would provide a good second characterization of an image. Thus, when an image and the corresponding $d\theta/dt$ are given as in Figure 2.17, a procedure is needed to extract $d\theta/dt$ of Figure 2.16c from $d\theta/dt$ of Figure 2.17.

This feature extraction procedure is also related to the problem of noise elimination. As is seen in Figure 2.20a, an image contour is normally distorted by noise. This noise appears in $d\theta/dt$ as a pulse sequence whose pulse heights are comparable with signal levels, but whose pulse intervals are much shorter than the intervals of signal pulses. Because of the high frequency characteristics, the noise may be eliminated by using a low pass filter. But the output of the filter becomes a smoothed continuous curve. Thus, a sharpening filter must follow the low pass filter to extract the pulse sequence from the smoothed curve.

The proposed sharpening filter locates pulse positions according to the location of the peaks of the smoothed curve and determines the pulse heights by the areas under the curve. Figure 2.20b shows an example of extracting a clean triangle from a noisy one.

SIMPLIFICATION OF IMAGE DESCRIPTION

In order to represent an image contour by a pulse sequence, the image contour must be constructed by line segments. However, it is not clear how many line segments are really needed to construct the essential structure of an image. Figure 2.21 shows an example in which an airplane is described by various pulse sequences. Figure 2.21a was extracted from a noisy airplane picture through the filters of Figure 2.20. Figure 2.21b was obtained by combining all convex pulses between adjacent concave pulses. Thus, the concave angles were preserved. Furthermore, making all convex angles $-180°$ and all concave angles $+90°$, the skeleton of Figure 2.21c results. Obviously, each stage simplifies the pulse sequence. If the objective is to distinguish an airplane from a missile, a very simplified pulse sequence might suffice. On the other hand, complex pulse sequences might be needed to classify different airplanes. The classification of various targets in terms of the complexity of pulse sequences is an area which merits additional investigation.

SIMILARITY MEASURE

The classification of targets requires a similarity measure or distance measure between two images. The amplitude characteristic of $F[d\theta/dt]$ has been selected for the following reasons:

1. The distance is invariant under various image variations such as rotation, shift, scale change, and the change of the starting point.

2. It is easy to eliminate the effect of noise by cutting off the high frequency components.

Letting $g_i(t) = d\theta_i/dt$ and $G_i(\omega) = F[g_i(t)]$, the distance between $g_1(t)$ and and $g_2(t)$ may be defined as

$$d^2(g_1, g_2) = \int_0^{\infty} [|G_1(\omega)| - |G_2(\omega)|]^2 d\omega$$

$$= \sum_{k=0}^{M} [|G_1(k)| - |G_2(k)|]^2 \quad . \tag{5}$$

The second line of Equation (5) is obtained because $g_i(t)$'s are always periodic with period 2π.

When images are characterized by relatively small numbers of line segments $g_i(t)$'s are pulse sequences. Thus, $|G_i(\omega)|$ and subsequently $d^2(g_1, g_2)$ can be easily obtained even by hand calculation resulting in a distance table among various targets with many viewing angles. When noise is added to an image contour, the noise in the $d\theta/dt$ domain may be characterized as a pulse sequence. The probability distribution of $d^2(g_i(t) + n(t), g_i(t))$ may then be studied in order to determine the amount of noise which can be tolerated during the classification of two specific targets. Figure 2.22 shows some of the typical noise patterns. It appears that any typical noise can be approximated by combinations of these patterns, thus providing a foundation on which to discuss the effects of noise on a theoretical basis. The problem of structure analysis of noisy images is presented in greater detail in Section 5.

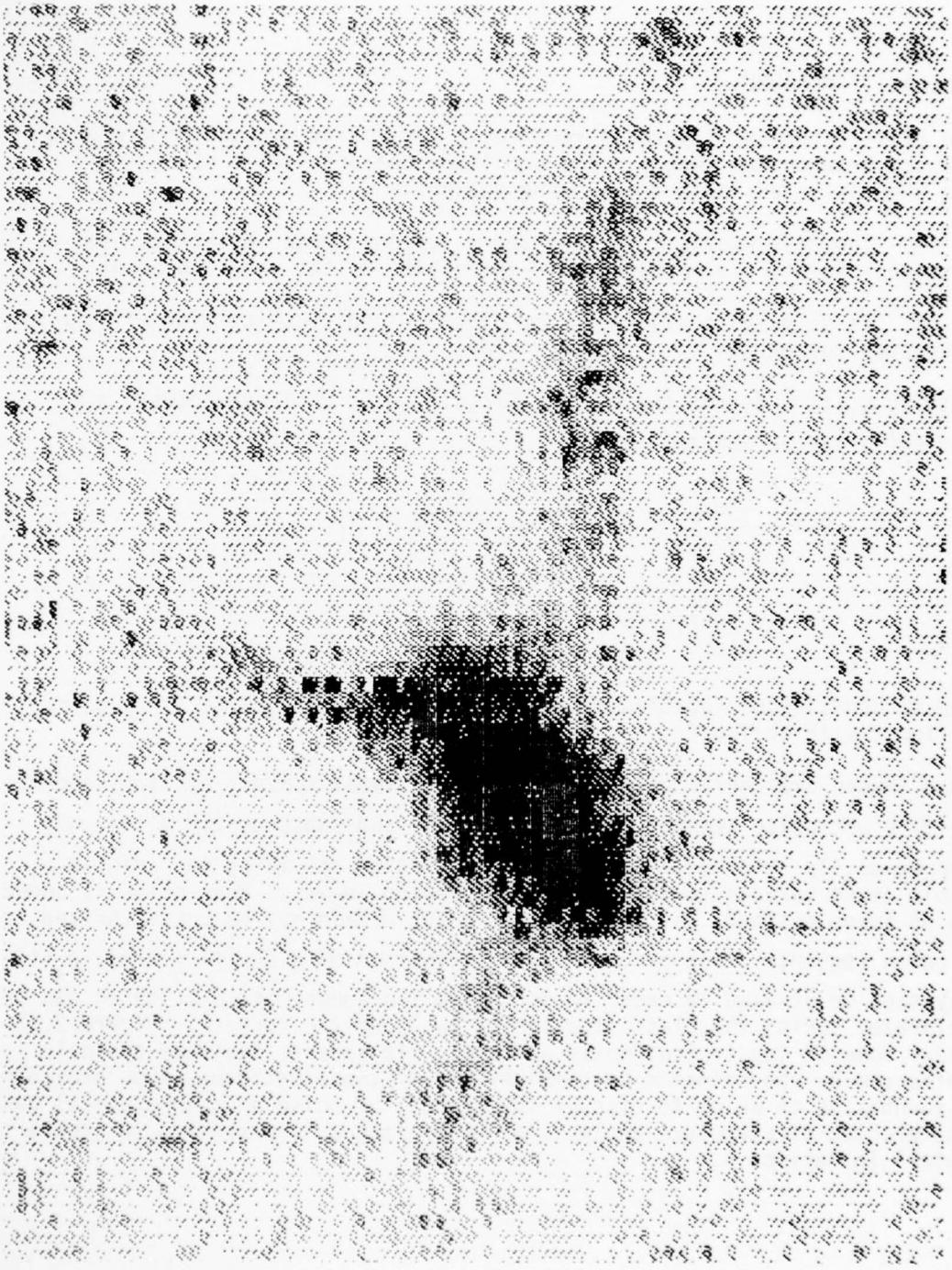


Figure 2.1. Original Plane

Figure 2.2. Median Filter Output from Figure 2.1

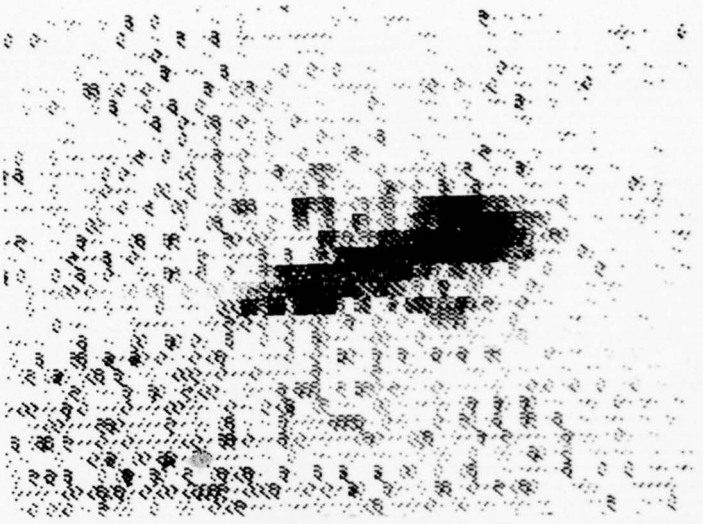


Figure 2.3. Cruise Missile Original

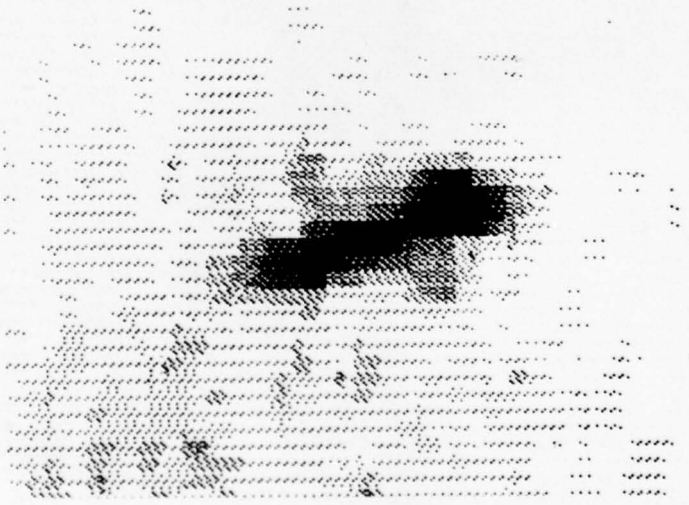


Figure 2.4. Cruise Missile Median Filtered

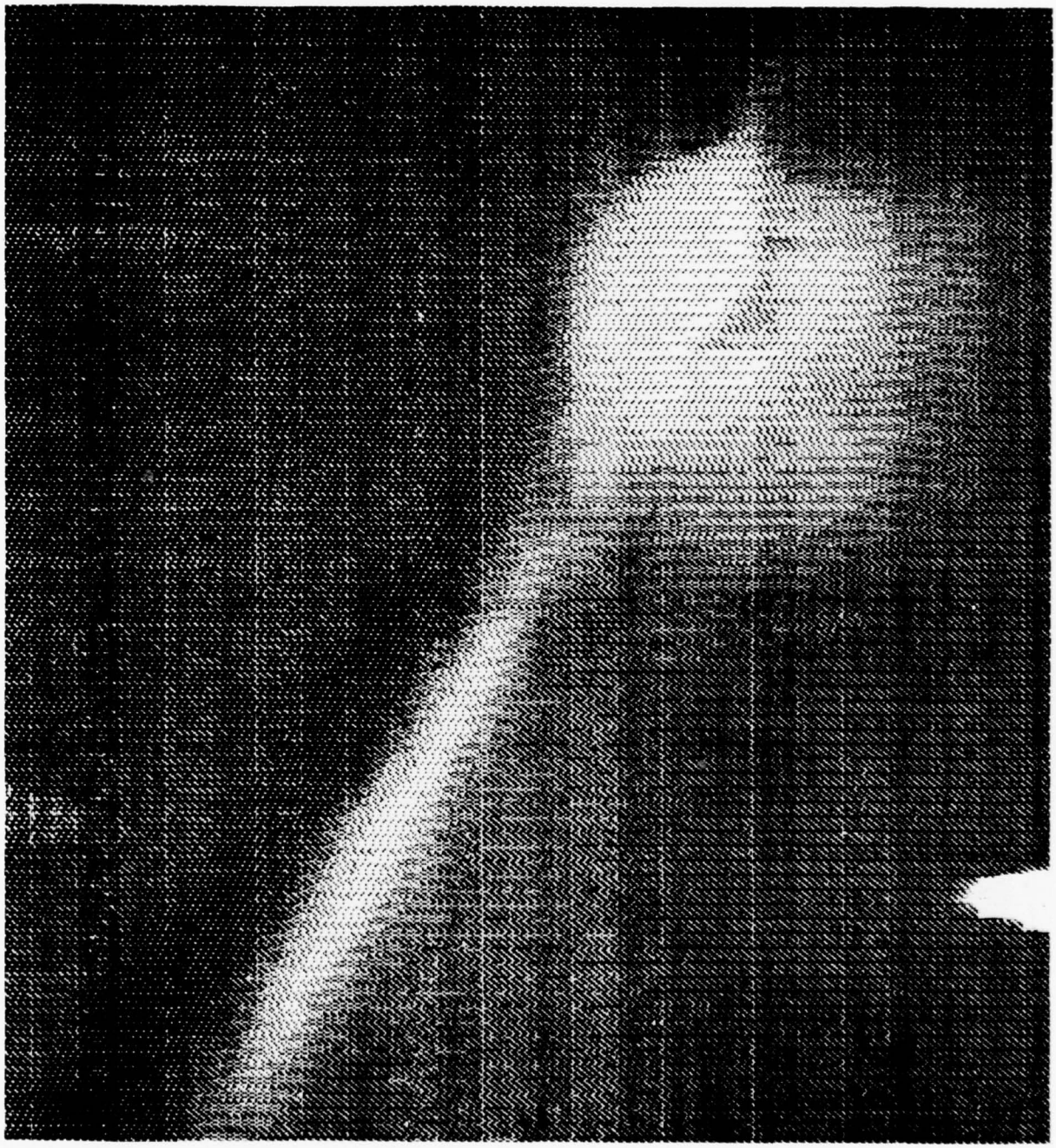


Figure 2.5. Missile Original

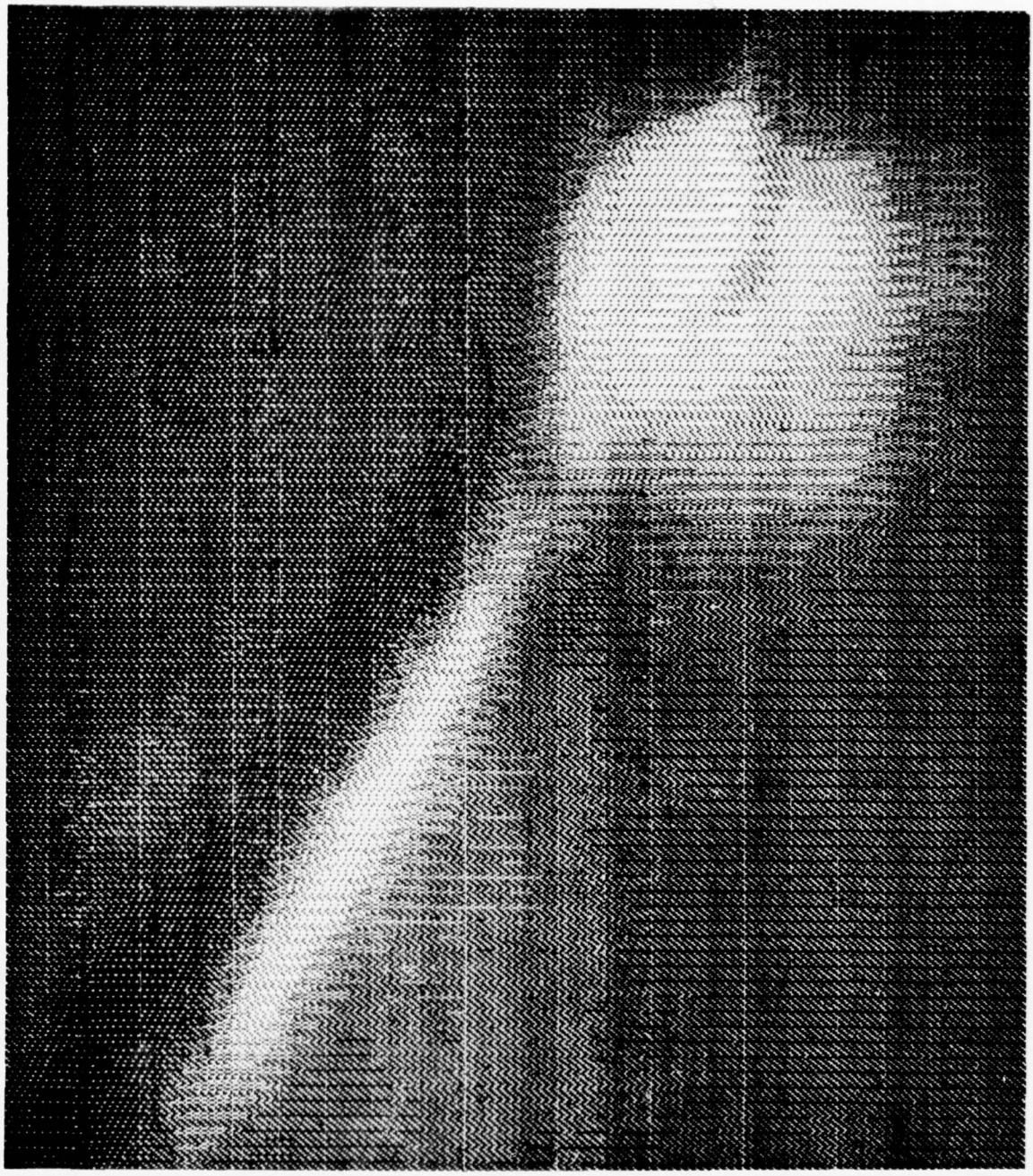


Figure 2.6. Median Filtered Missile

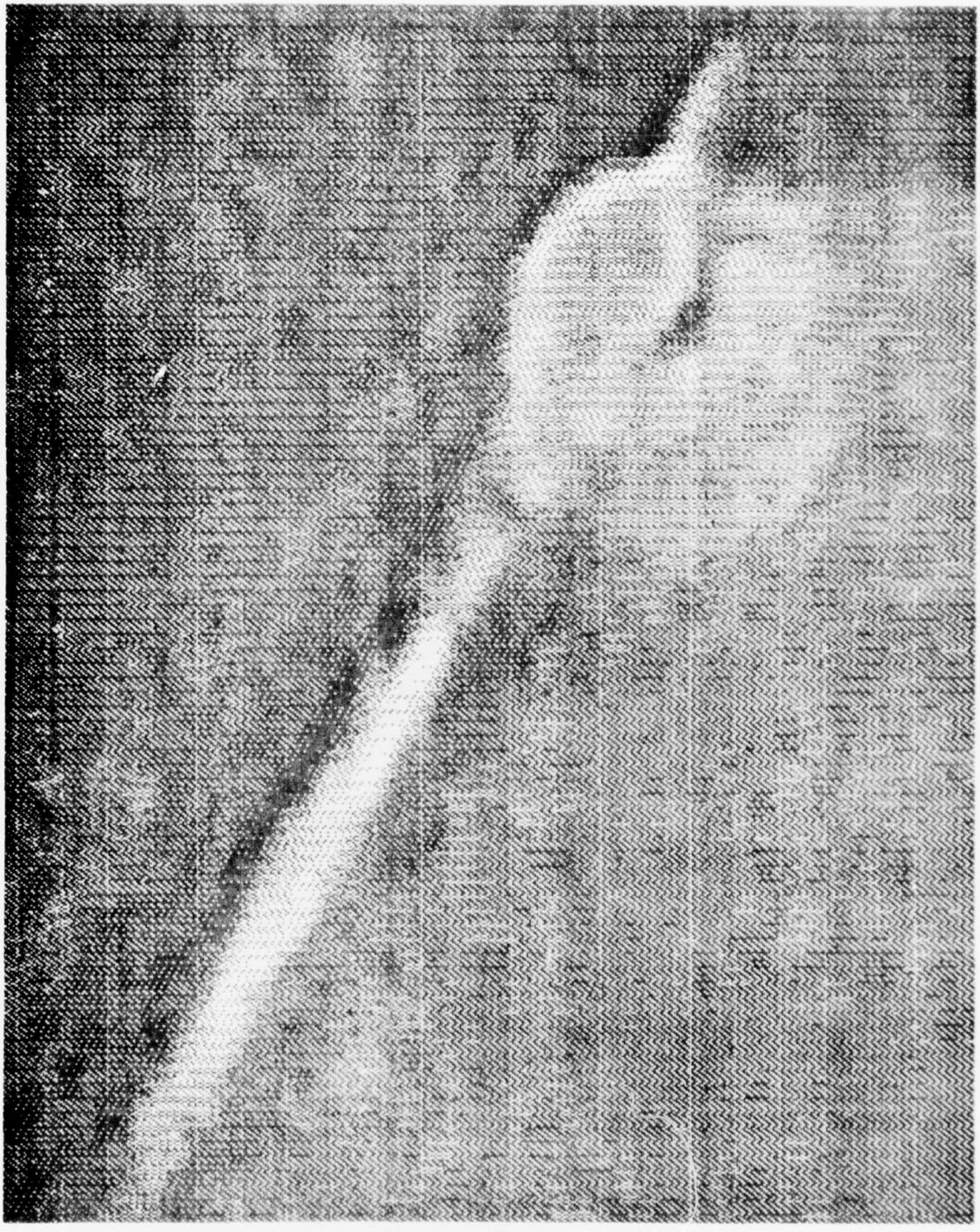


Figure 2.7. Effect of Human Visual Filter Operating on Figure 2.6

Figure 2.8. Result of Averaging Figure 2.2 over a 5x3 Window

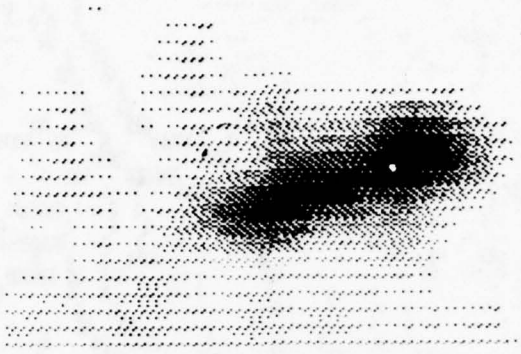


Figure 2.9. Result of Averaging Figure 2.4 over a 5x3 Window

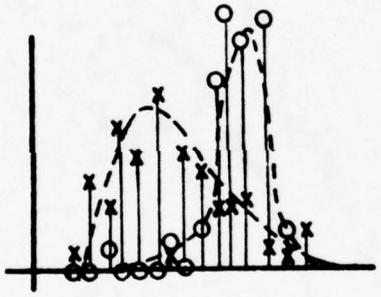


Figure 2.10. Two Histograms from Two Windows

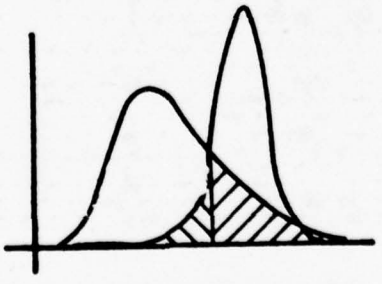


Figure 2.11. Smoothed Histograms and Resulting Overlap Error

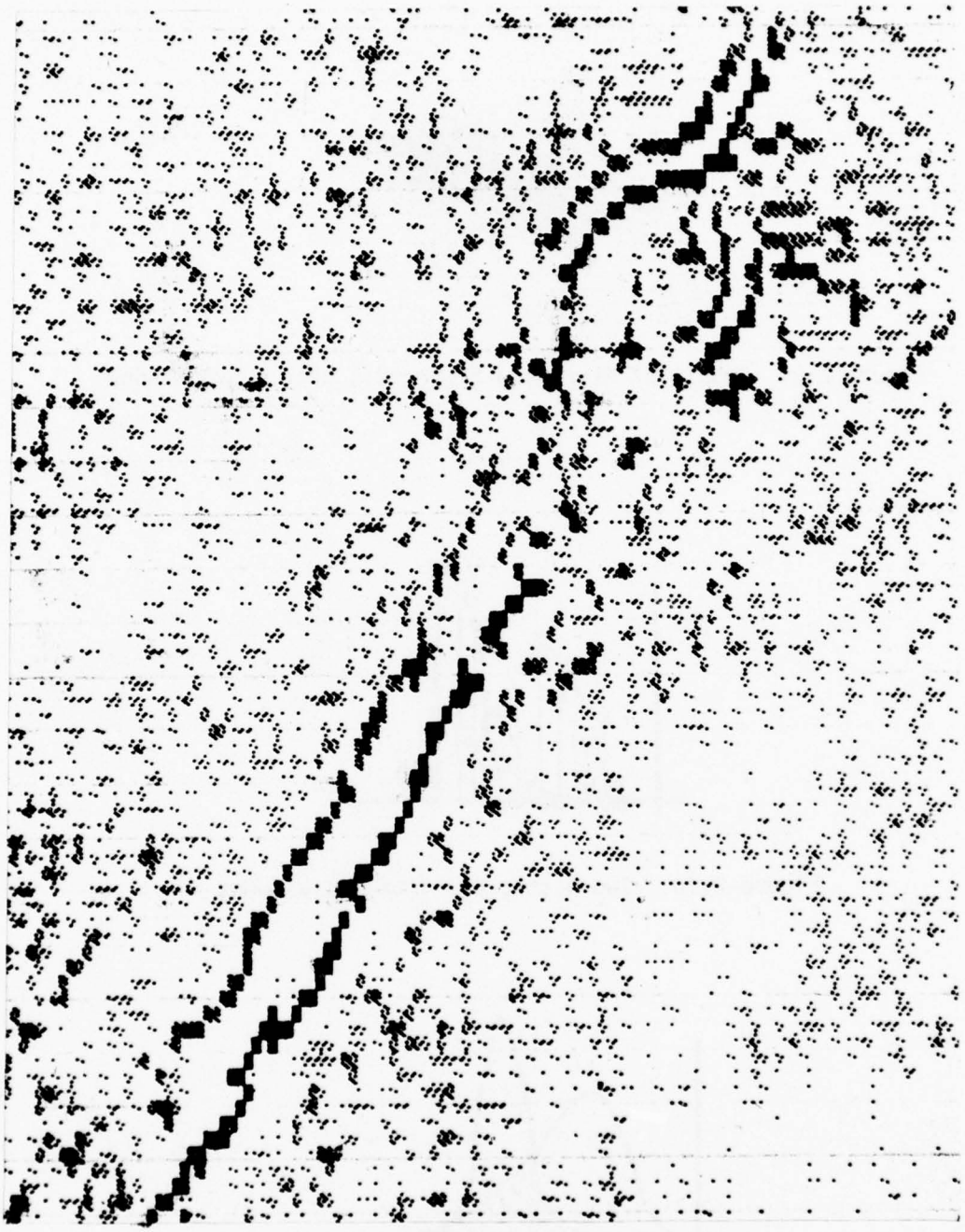


Figure 2.12. Local Extrema on Figure 2.7
(Darker points are larger extrema)

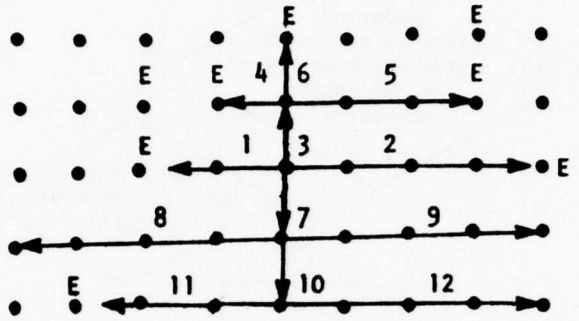


Figure 2.13. Example of Edge Points Controlling Window Shape

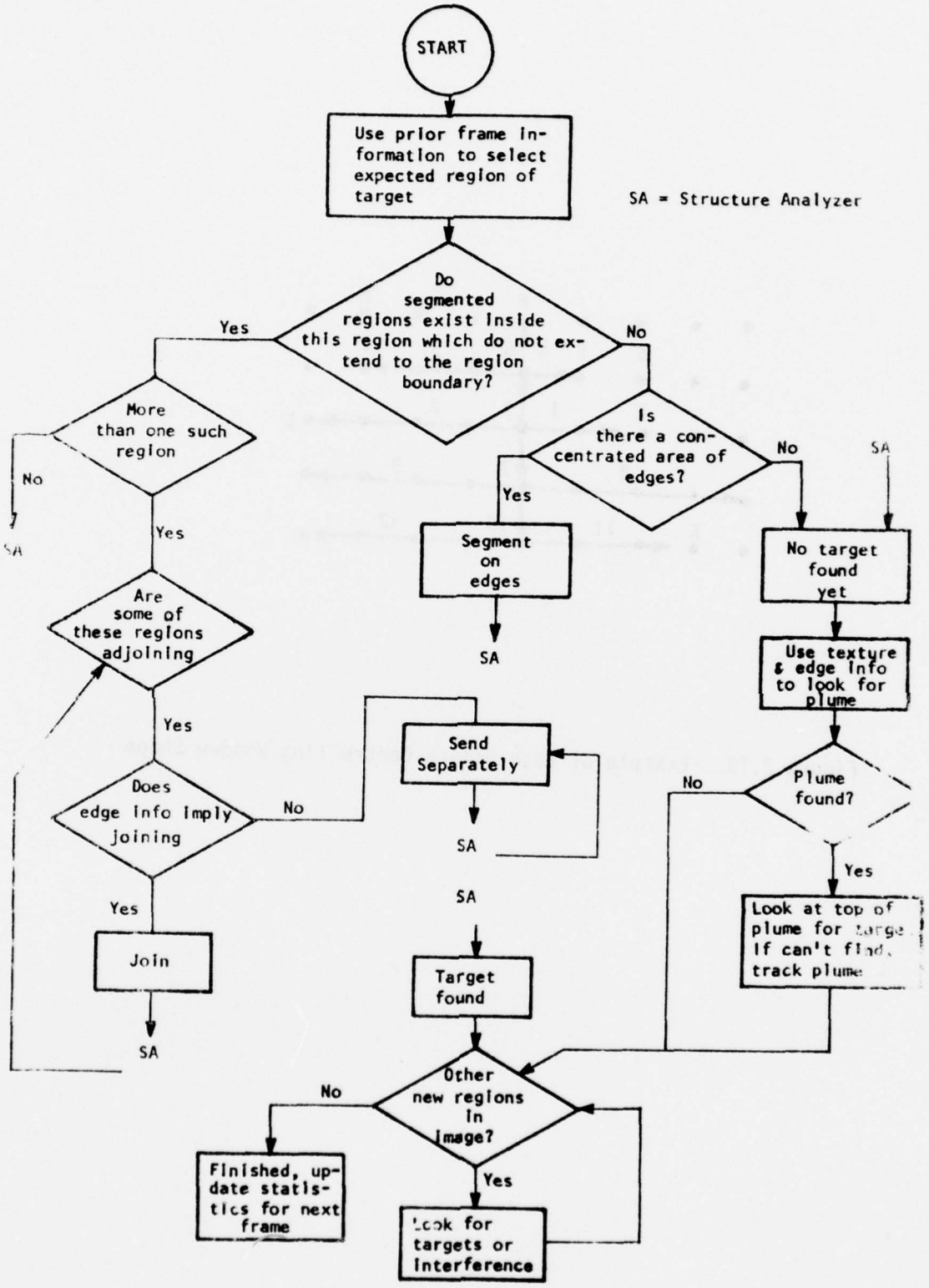


Figure 2.14. Target Extraction Protocol

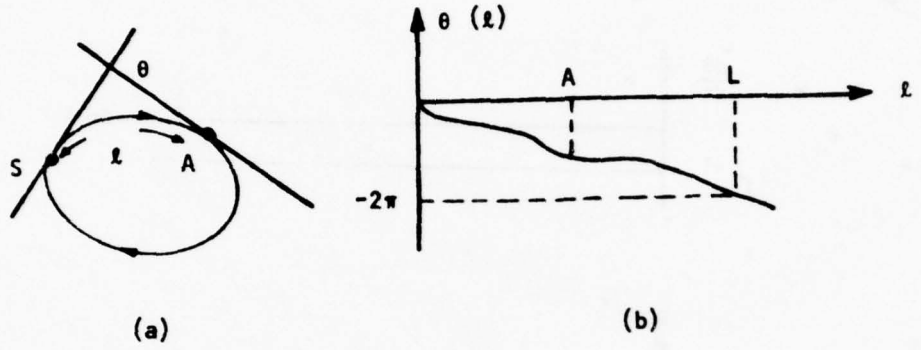


Figure 2.15. An Image Contour and its One-dimensional Representation

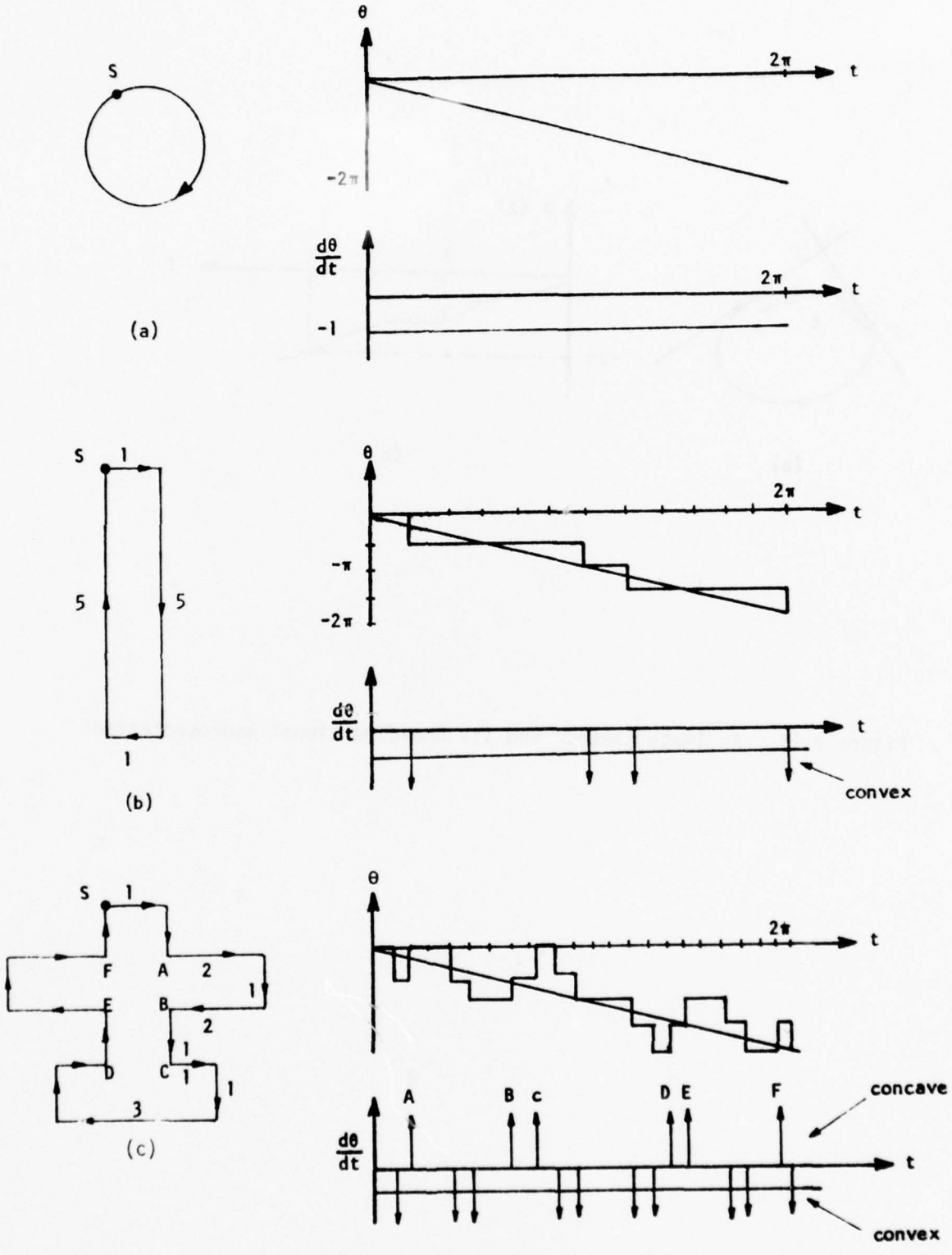


Figure 2.16. θ and dθ/dt for Various Images

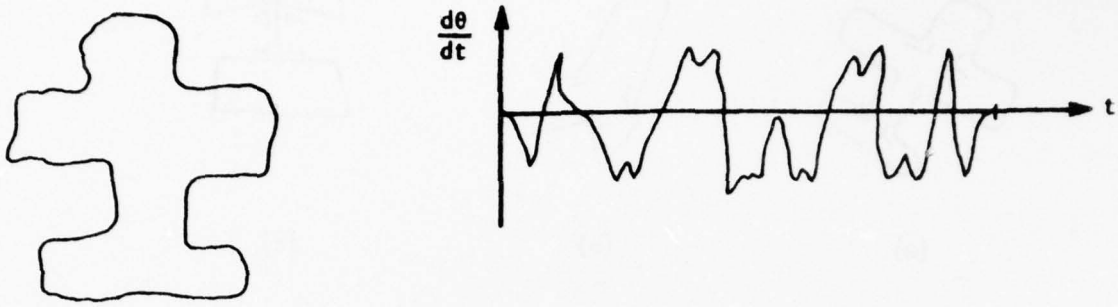


Figure 2.17. A Smoothed Contour and its dθ/dt

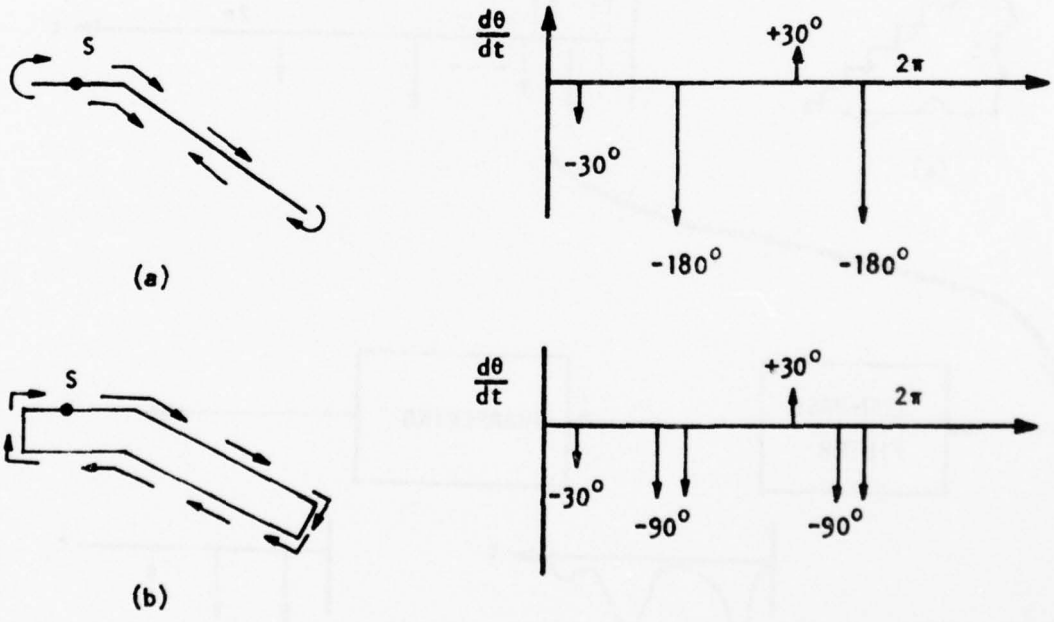


Figure 2.18. The Contour of an Unclosed Line

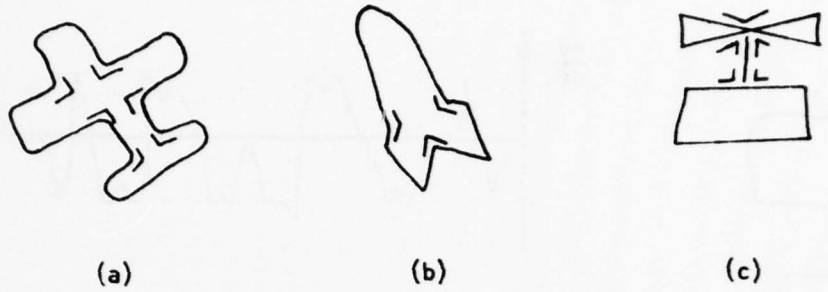


Figure 2.19. Concave Angles of Various Targets

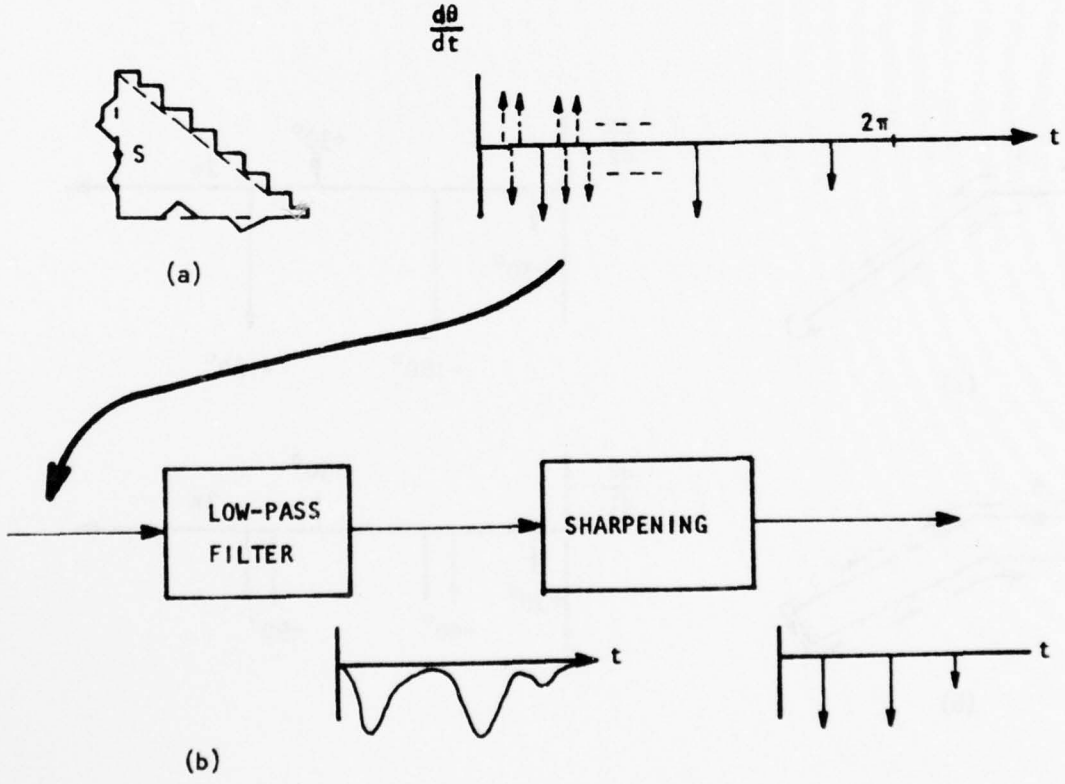


Figure 2.20. Extraction of Line Segments from a Noisy Triangle

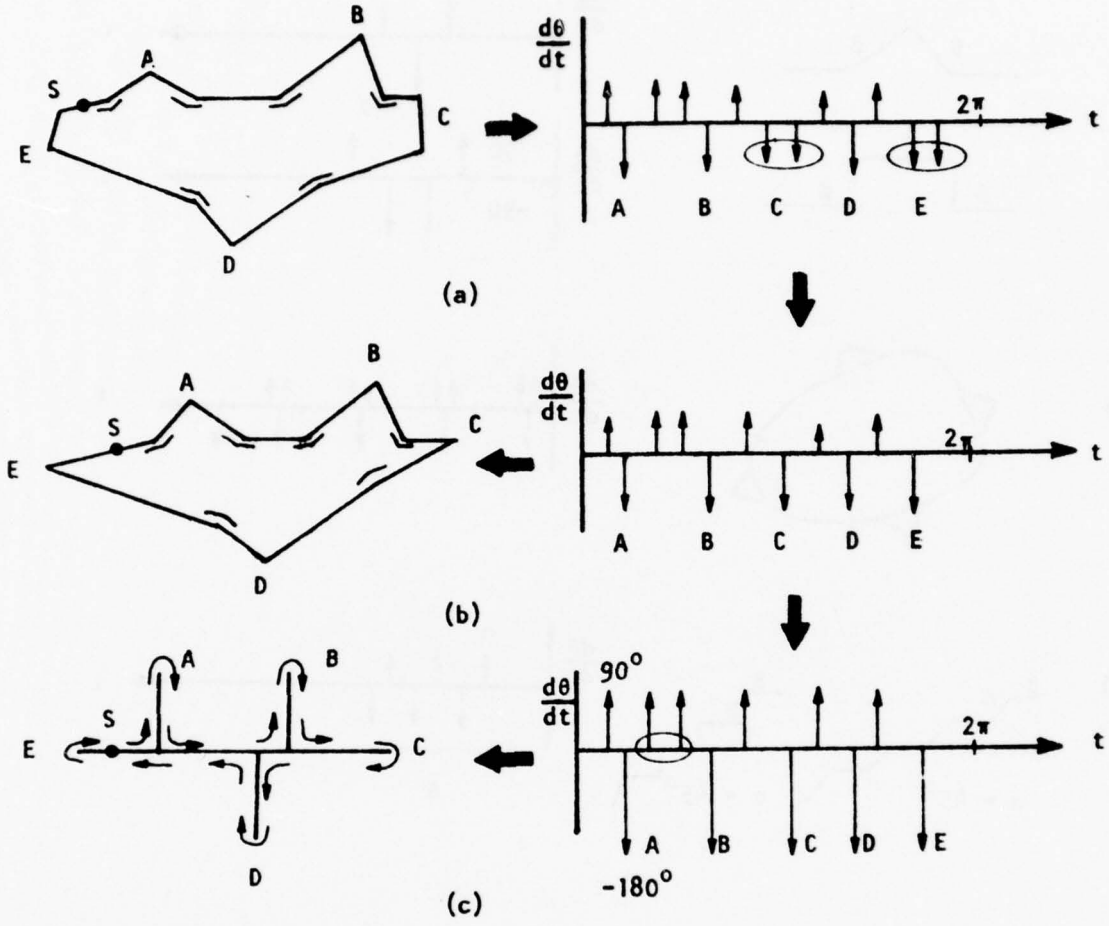


Figure 2.21. Various Representations of an Airplane

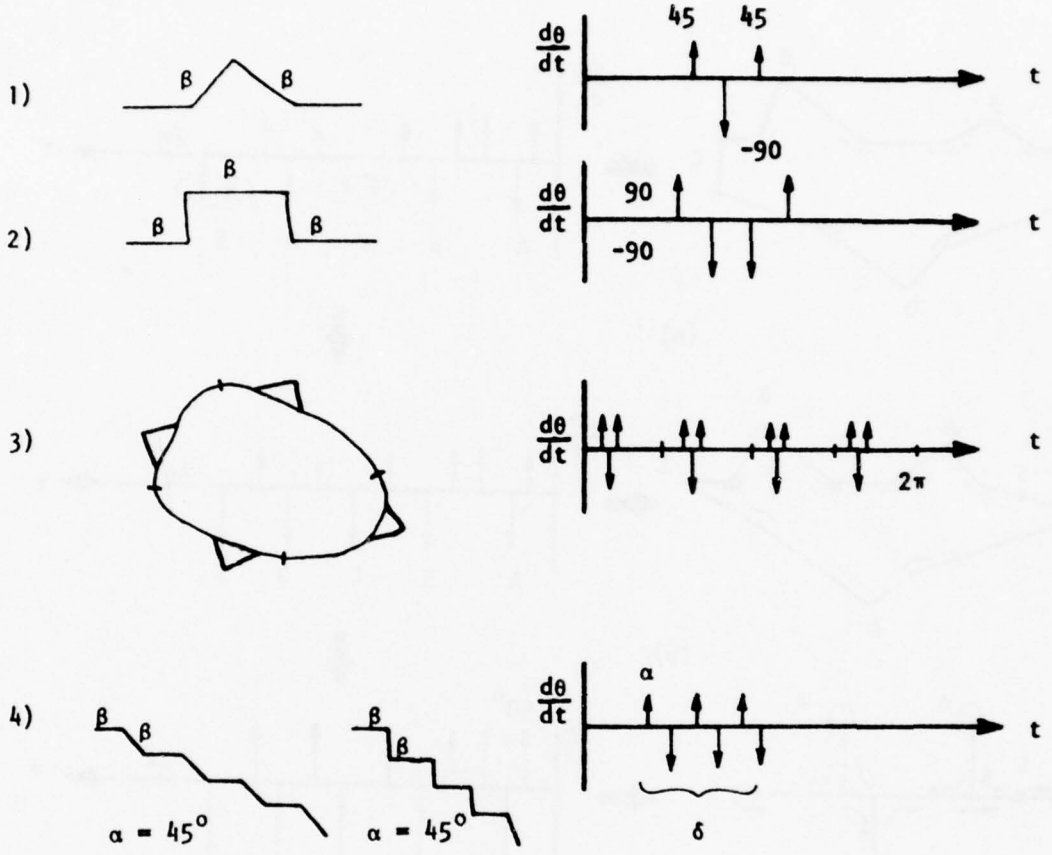


Figure 2.22. Noise Patterns

SECTION 3

PREPROCESSING AND SEGMENTATION ON THE PDP-11

Several programs have been written and some preliminary results derived on the PDP computer using digitized video data. Some of the results were reported under Segmentation in Section 2. The preprocessing programs written are MEDIAN (median filtering), MAX256 and MAX512 (local extrema detection), AVG (window averaging), and HVS256 (HVS filtering). Actual source codes are given in Section 8.

The MEDIAN and AVG programs use a collection of linked FORTRAN subroutines that allow processing on any size window over any size picture. This choice can be made at run time. The MAX256, MAX512, and HVS256 were developed separately and have compiled picture and window sizes. Following is a discussion of each program separately.

MEDIAN

MEDIAN outputs a picture in which each point is replaced by the median of all points in a designated size window surrounding it. Actually, the program requests the filter order, which is the point to be selected after sorting all window points (for the median of a 3 x 3 window, select order = 5). The input file, output file, picture size, window size, and filter order are all requested at run time.

AVG

Similar to MEDIAN except all points in the designated size window are averaged to determine each output point.

HVS256

Outputs a picture which is the convolution of the input picture and the two-dimension HVS filter function shown in Table 2.2 of Section 2. The output is truncated between 0 and 255 (8-bits per point). The input and

output files are requested at run time. The input picture must have 256 pixels per row. The number of rows does not matter.

MAX256

Locates local extrema in the input picture and outputs their size as an output picture. See Local Extrema Information in Section 2 for a description of this process. The sizes are multiplied by 2 and truncated at 255 (8 bits). The low order 2 bits are used to indicate whether the extremum is a maximum or minimum and whether it is a horizontal or vertical extremum. The window size is presently compiled to be 12 vertical and 24 horizontal points in each direction from the center point. The input and output files are requested at run time. The input picture must have 256 pixels per row. The number of rows does not matter.

MAX512

Same as MAX256, except the input picture must have 512 pixels per row.

PROCESSING RESULTS

Sample processing results that were not used in Section 2 are now presented. Appropriate comments are included in the captions. Figures 3.1 through 3.17 are half tone gray level displays printed on the Printronix line printer using a 3 x 7 dot matrix to produce 22 gray levels.

Figure 3.1. Missile 1, View 1, Before Launch

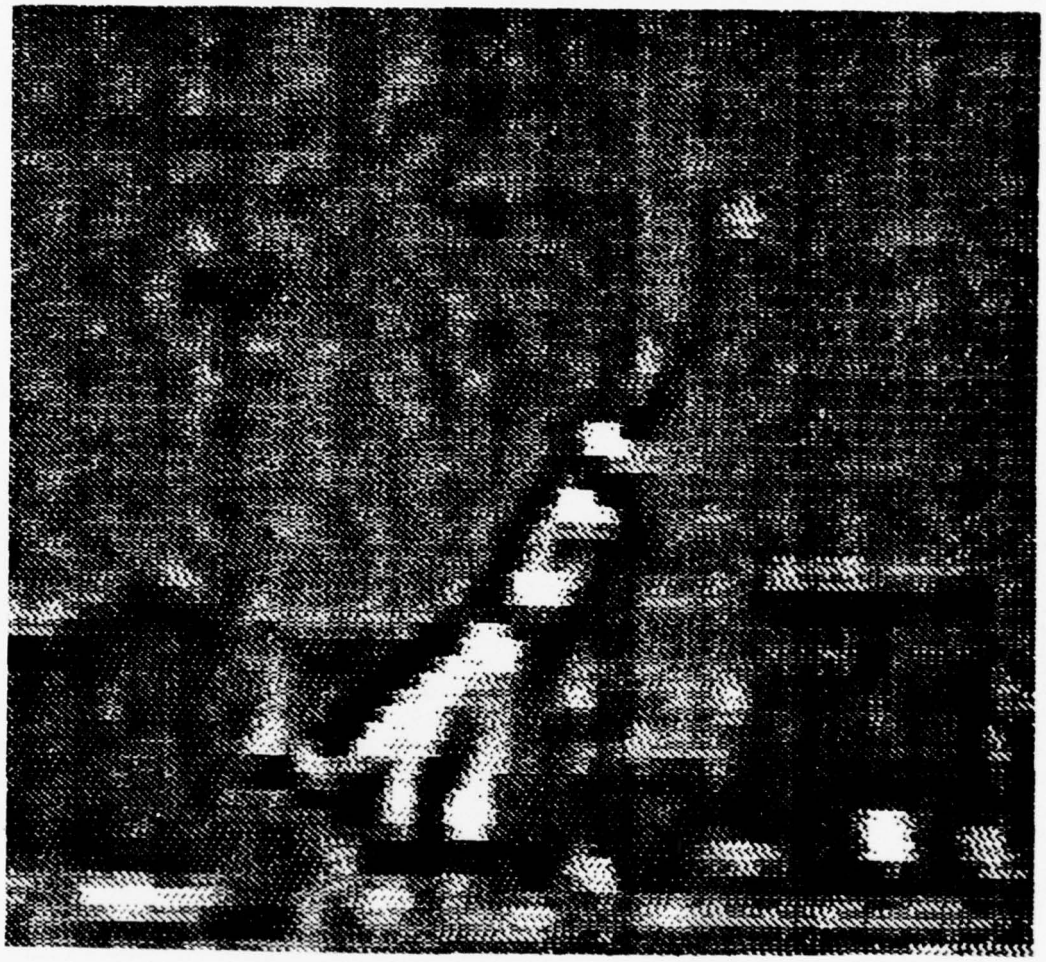


Figure 3.2. Human Visual Filter Operation on Figure 3.1

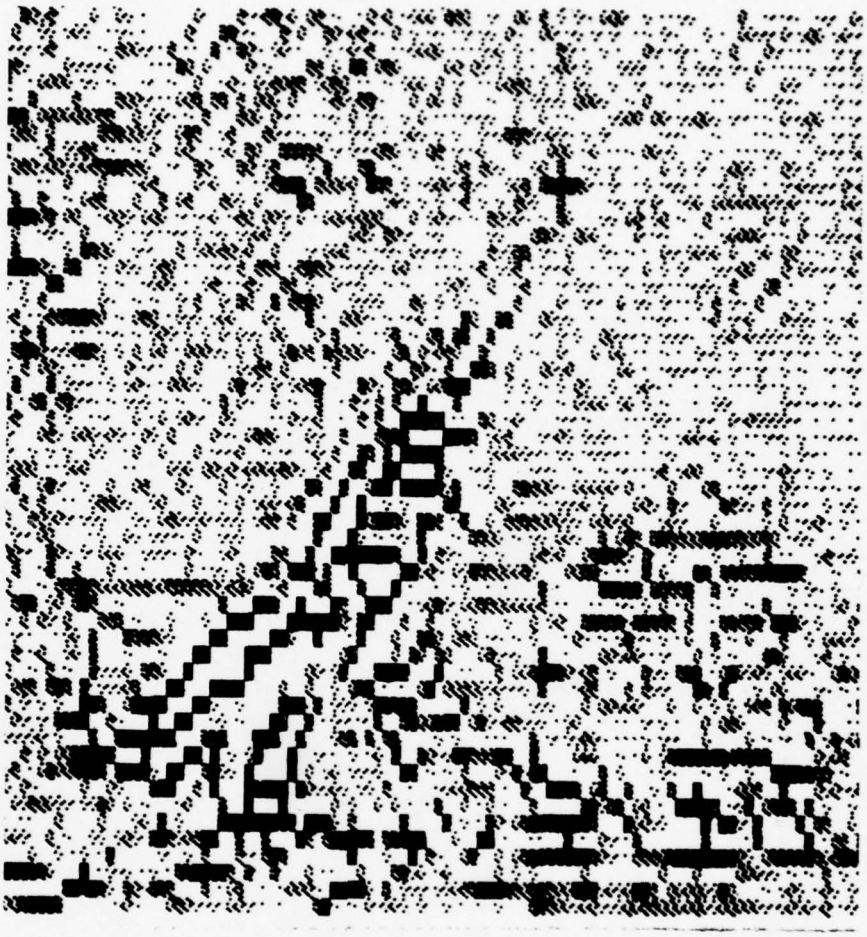


Figure 3.3. Local Extrema Detected on Figure 3.2
(Larger size extrema are indicated blacker.
Video invert used on display program.)

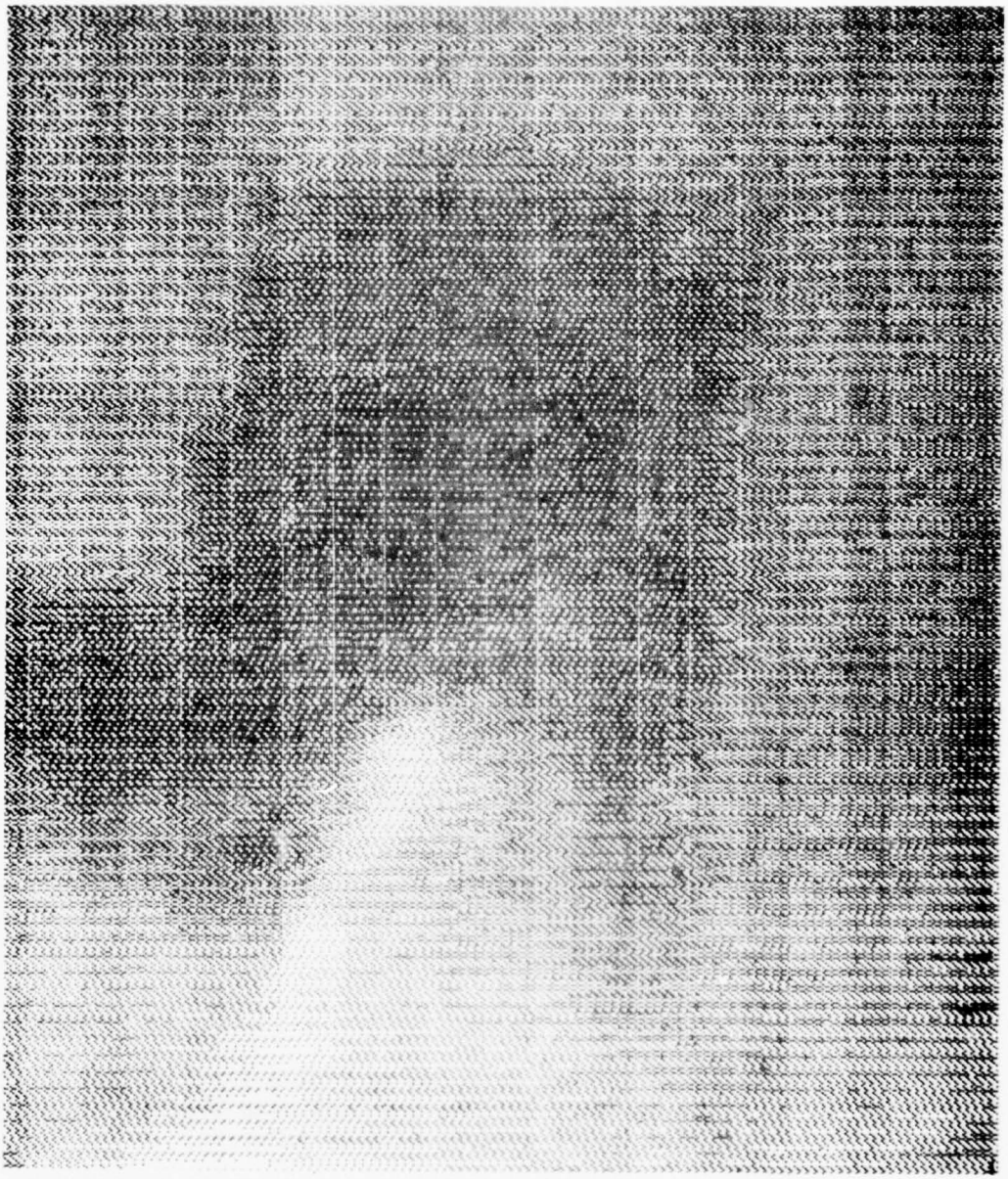


Figure 3.4. Missile 1, View 2
(Smoke obscures missile)

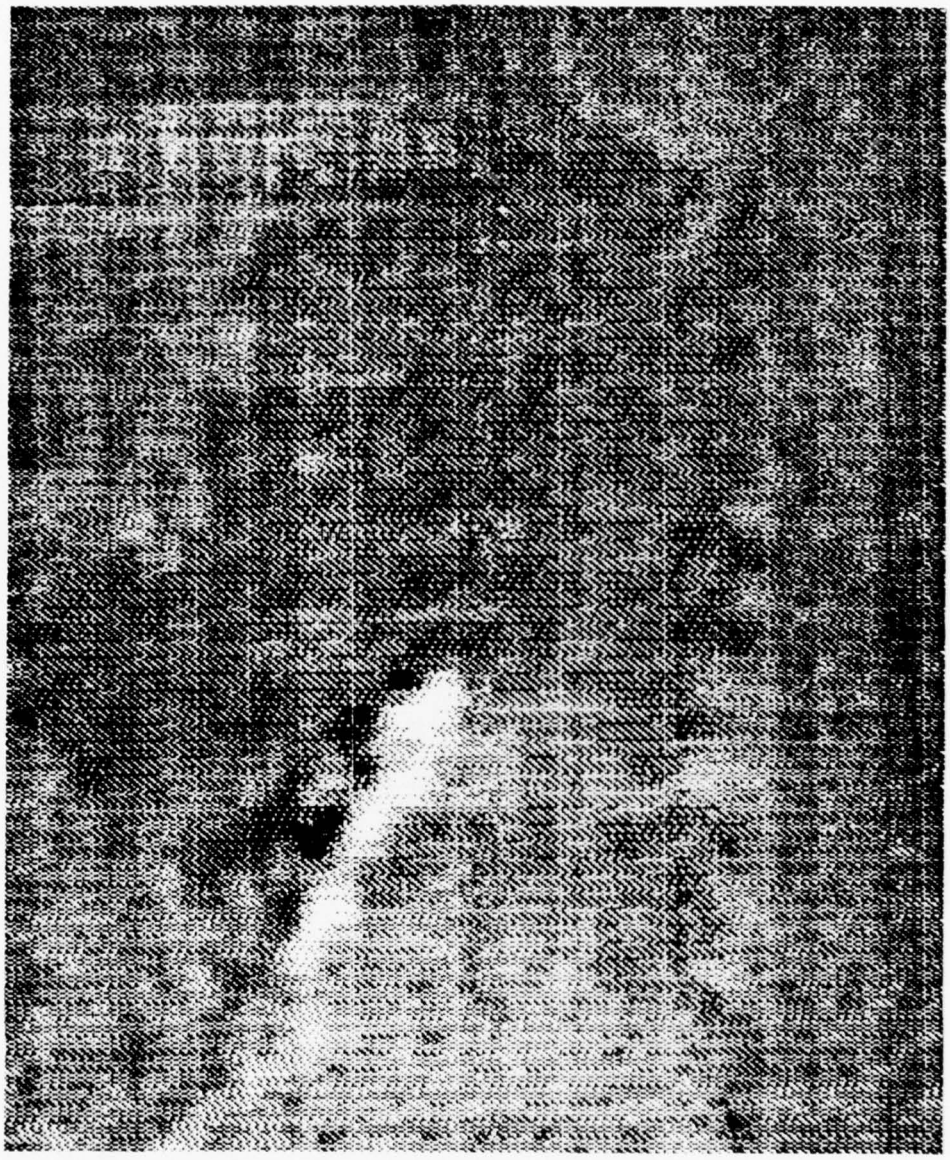


Figure 3.5. Human Visual System Filter of Figure 5.4

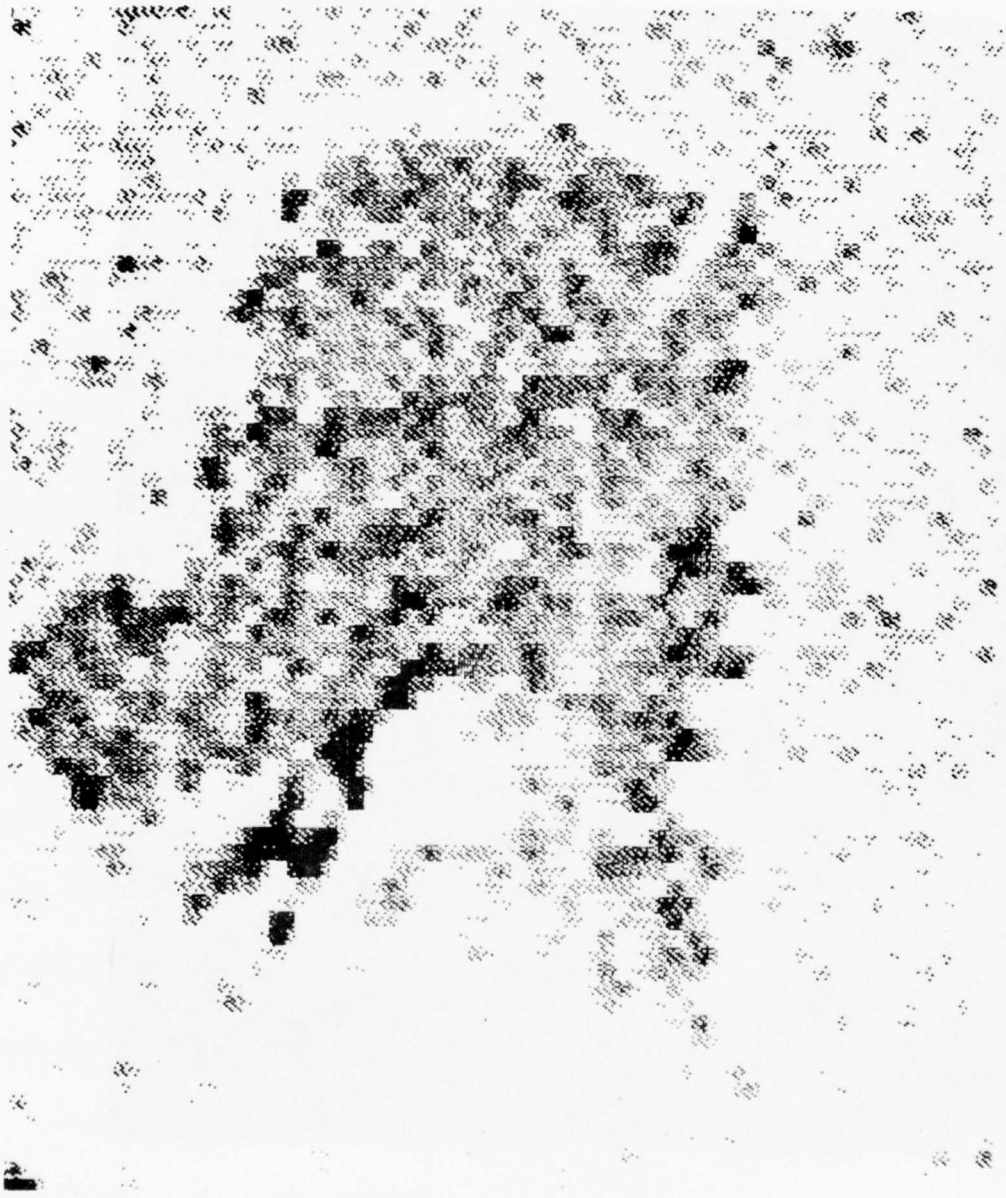


Figure 3.6. Expanded Contrast of Figure 3.5
(Missile tip is barely visible)

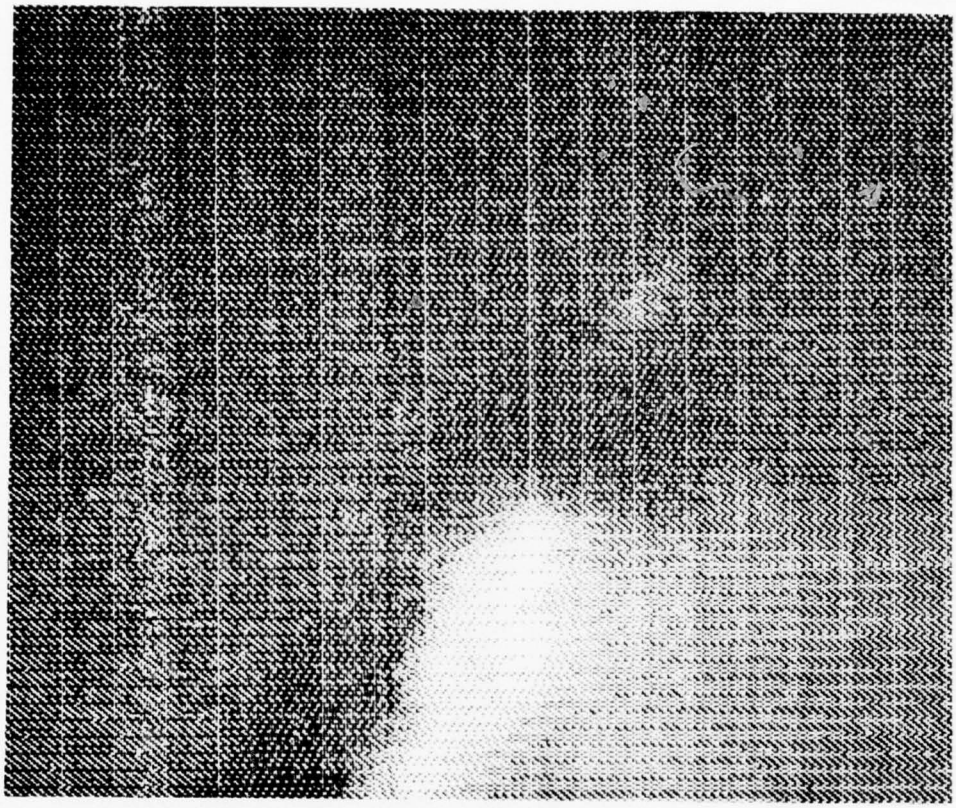


Figure 3.7. Missile 1, View 3

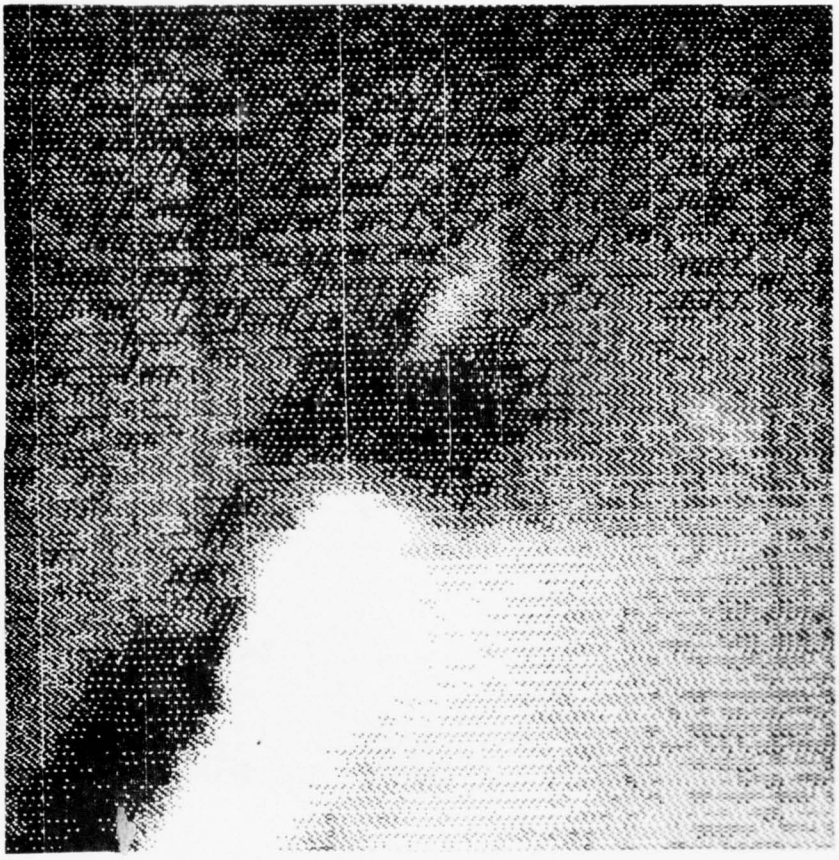


Figure 3.8. Expanded Contrast of Figure 3.7 (To see how visible missile is in original)

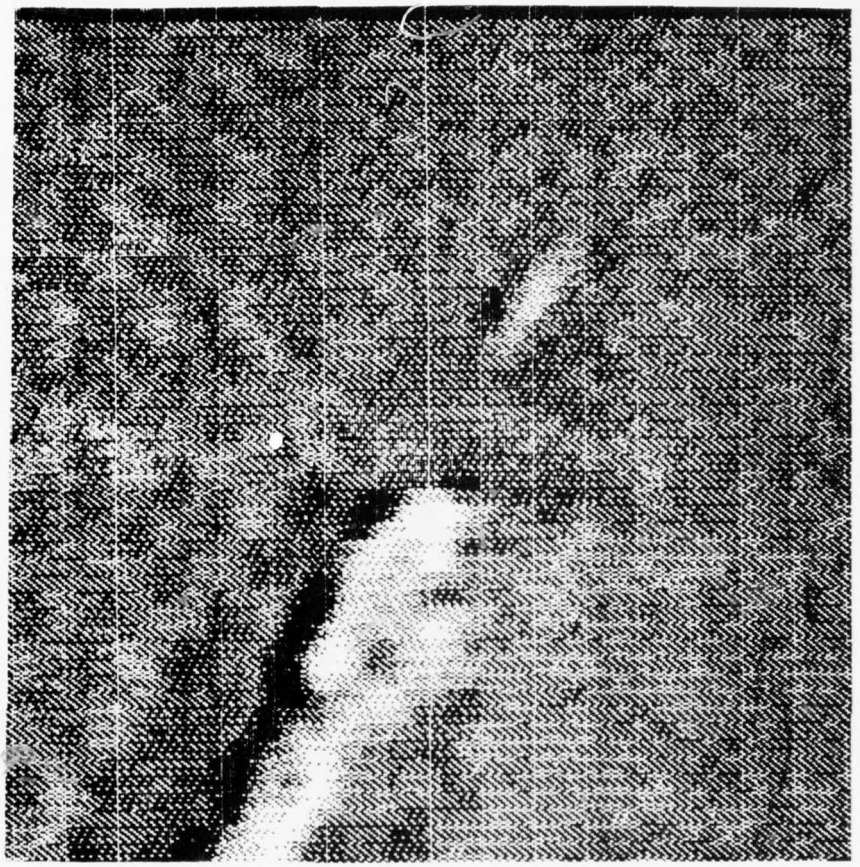


Figure 3.9. Human Visual Filter of Figure 3.7

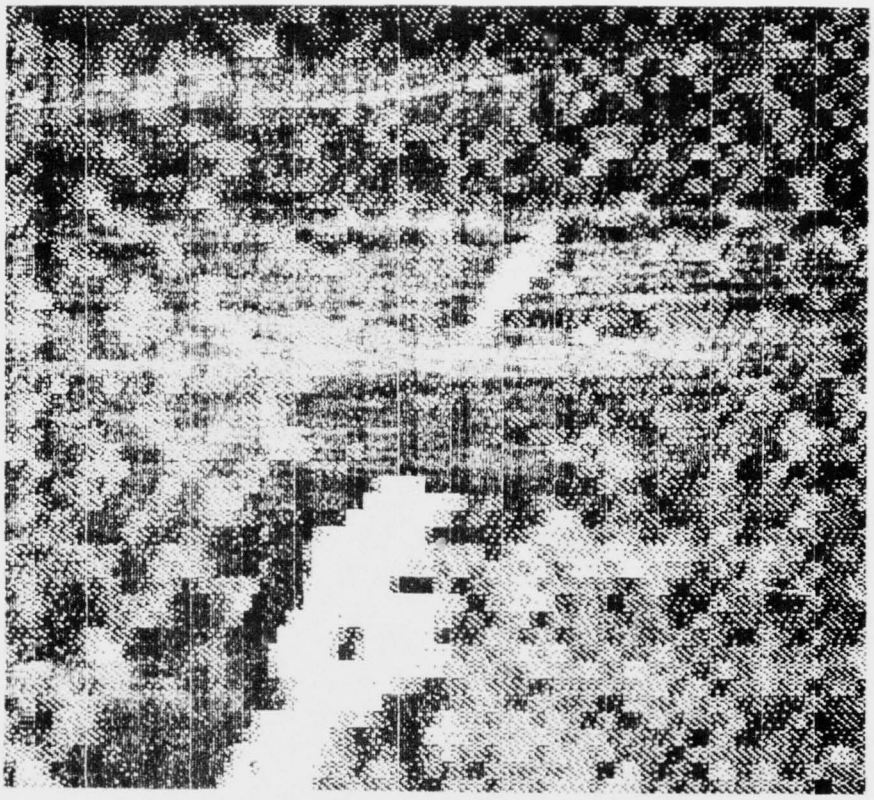


Figure 3.10. Contrast Expanded Version of Figure 3.9

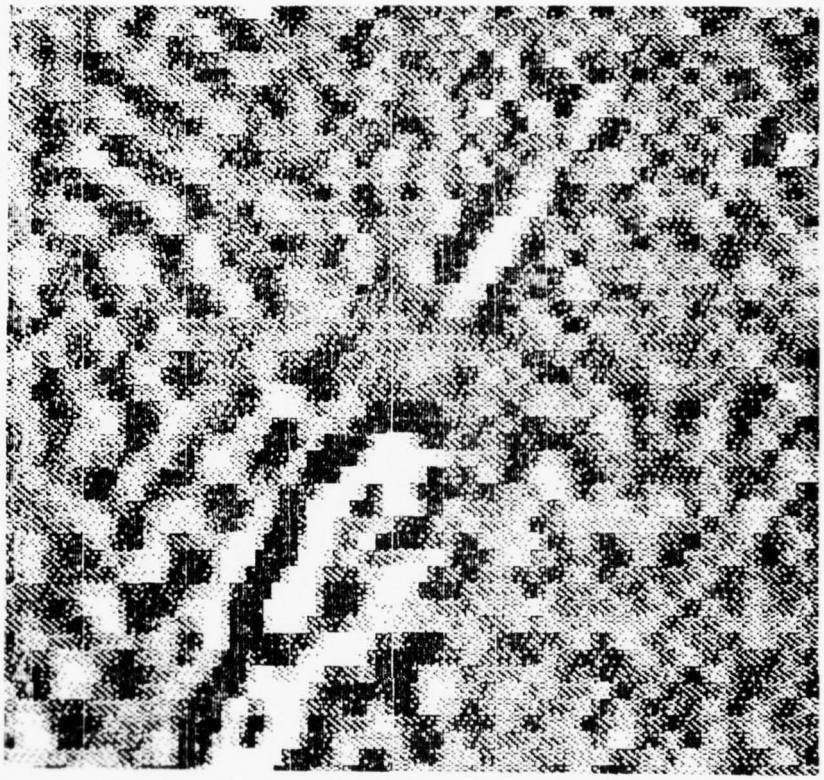


Figure 3.11. Effect of Sending Missile 1, View 3 through Human Visual System Filter Twice

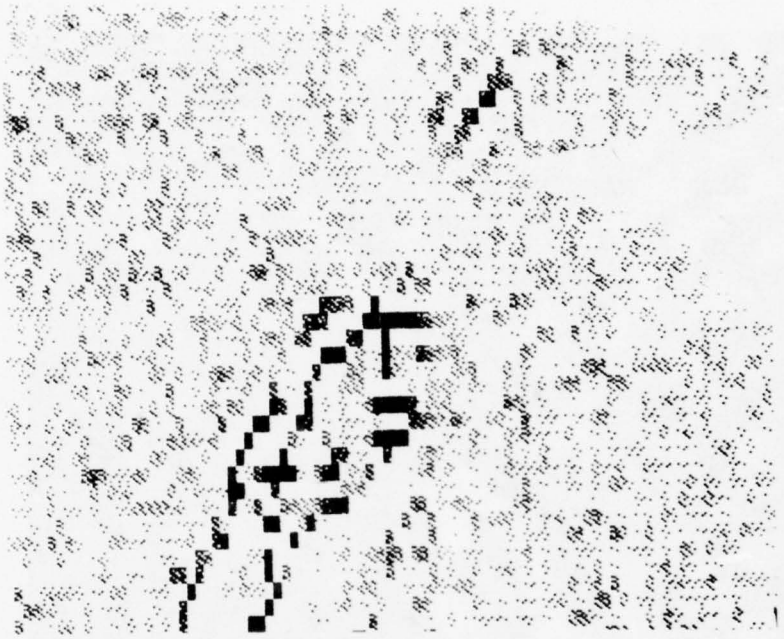


Figure 3.12. Local Extrema from Figure 3.9

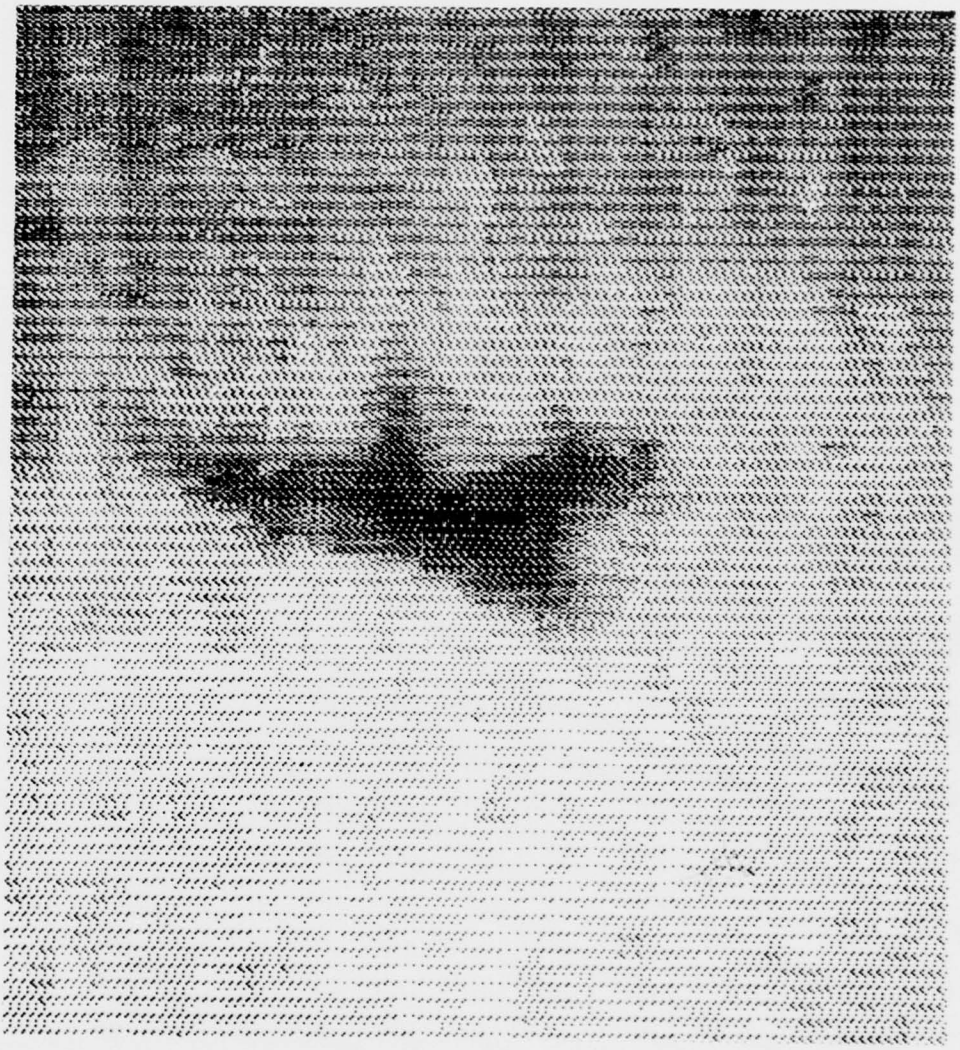


Figure 3.13. Plane 2, View 2

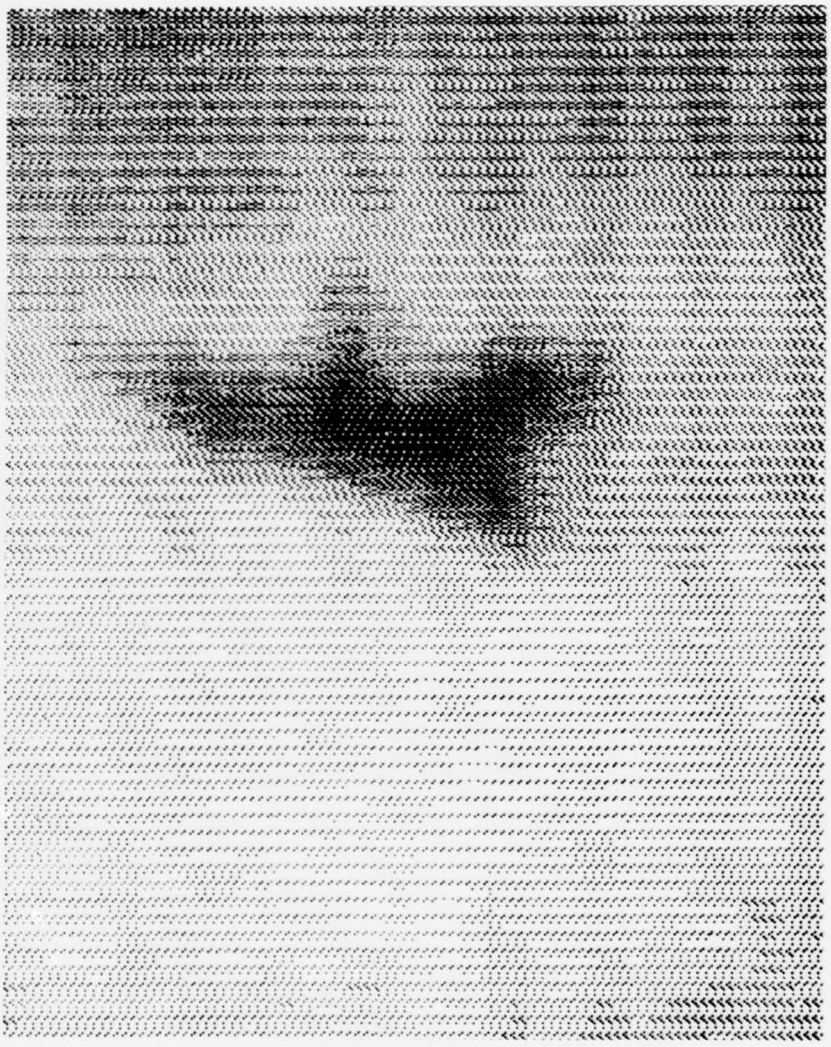


Figure 3.14. Plane 2, View 2
(Median filtered 3x3 window)

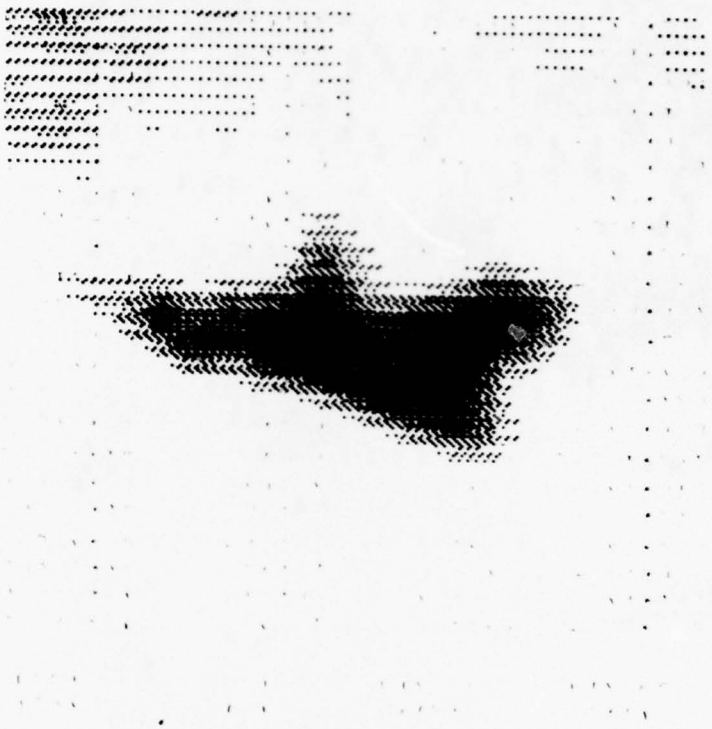


Figure 3.15. Plane 2, View 2
(Median filtered 3x3 window,
then averaged over 5x3 window)

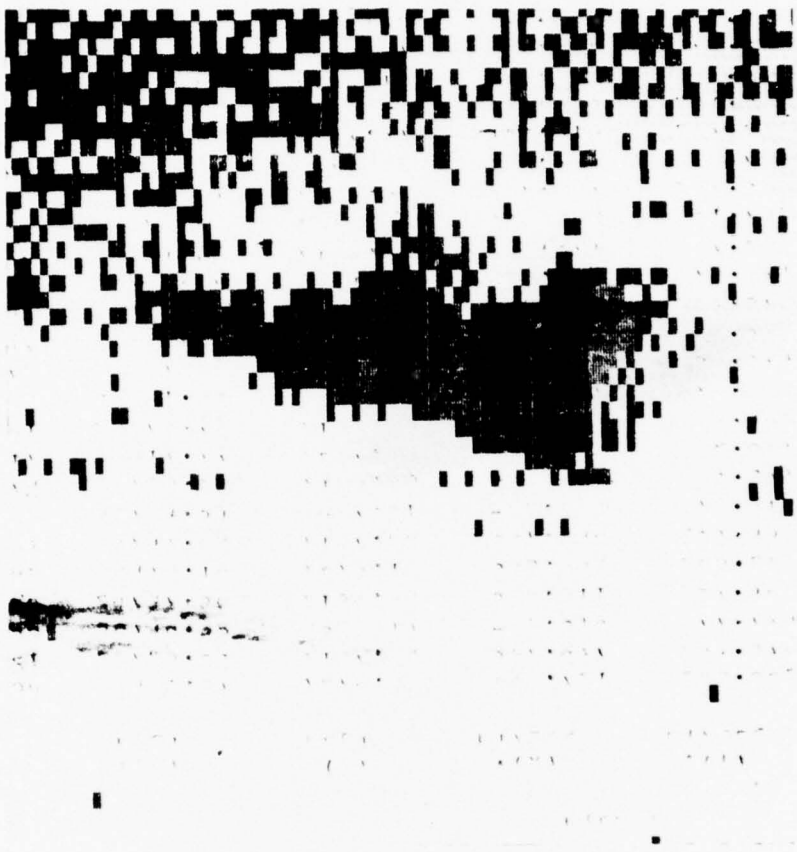


Figure 3.16. Threshold on Figure 3.13

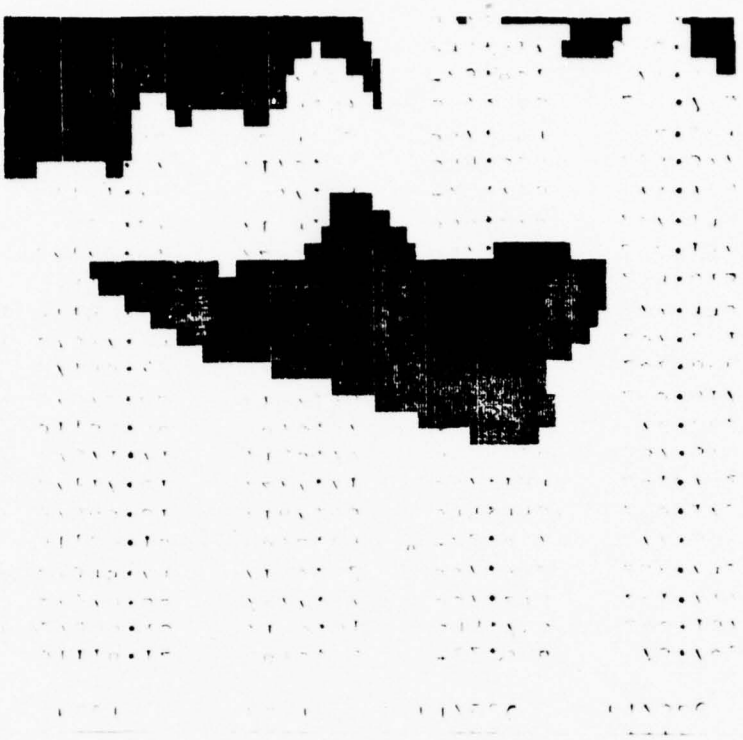


Figure 3.17. Threshold on Figure 3.15

SECTION 4

SEGMENTATION AND STRUCTURE ANALYSIS OF THE UNIVAC 1108

The RT11 preprocessing programs used on the PDP 11/35 system are used to produce binary images which contain potential targets for tracking. Processing programs written for the 1108 are used primarily to analyze the structure of the binary images. A tape copying routine is also included in the 1108 software package. Printouts of the source codes are included in Section 8 with some documentation. A brief description of the major programs is presented in this section.

IMAGAN (IMAGE AND NOISE SIMULATION)

This program adds a noisy Markov field to an input image boundary and generates a noisy image. It then performs a thresholding operation and prints out the resulting binary image. Finally, it traces the boundary of the noisy image and stores it for additional analysis.

PROG2 (STRUCTURE ANALYSIS OF IMAGE BOUNDARY)

This program generates a one-dimensional angle function $\theta(\ell)$ from the input boundary points of a noisy image obtained from IMAGAN or PICTPR. It then extracts the pulse sequence $\dot{\theta}(\ell)$ from $\theta(\ell)$ and runs it through a low pass Gaussian filter with a specified cutoff frequency to remove the noise pulses. Finally, it calculates the Fourier coefficients of the filtered angle pulse function.

PICTPR (TARGET BOUNDARY EXTRACTION FROM REAL IMAGE DATA)

This program reads a specified window of data from a picture file stored on 7-track magnetic tape. It then selects a specified number of gray level thresholds, prints out the picture, and finally generates an edge contour of the target which can be analyzed using PROG2.

COPY (TAPE COPY ROUTINE)

The 8-bit pixels from picture data files are copied onto 7-track tape using the routine MAG256 or MAG512. Copies of these 7-track data tapes are made using the program COPY on the UNIVAC 1108. The data is read from the data tape and then packed onto the copy tape using a specified format. The user specifies the number of records per file, the number of words per record (assuming 36-bit words), and the number of files to be copied.

SECTION 5

STRUCTURAL ANALYSIS OF NOISY IMAGES USING ONE-DIMENSIONAL CONTOUR DESCRIPTORS

INTRODUCTION

The problem of representing a simple closed contour by a one-dimensional function has been studied in the literature with some interesting results. The Fourier descriptors, first introduced by Cosgriff[11], and further reported on in an excellent paper by Zahn and Roskies[12], is one such method. Zahn proposes a normalized function $\phi(t)$, in which the tangential angle is considered as a function of the contour length and then normalized. In this paper, we restrict the class of contours to polygons, which, as Zahn notes, arise naturally in the study of digitized pictures. The problem of extracting such boundary contours has been well studied[13,14,15,16]. For our purposes, we assume the contour has already been extracted, and discuss various characteristics of these contours in terms of the derivative of $\phi(t)$. By taking the derivative of $\phi(t)$, we show that a polygonal contour may be represented by a sequence of impulse functions. These impulses have a simple relationship to the convexity or concavity of the polygonal angles. This relationship facilitates the defining of a simple complexity measure, which is closely related to the concavity of the contour. We also discuss a similar pulse representation for the skeleton. Next, a slightly modified version of Zahn's similarity measure is proposed in terms of the Fourier descriptors corresponding to $\dot{\phi}(t)$. The effects of noise in the contour is discussed in terms of this measure and a technique for extracting a noise free contour from a noisy one is proposed. Finally, several examples are given throughout the paper.

11. R. L. Cosgriff, "Identification of Shape," Ohio State University Research Foundation, Columbia, Rep. 820-11, ASTIA AD 254792, December 1960.

12. Zahn and Roskies, pp. 269-281.

13. C. T. Zahn, "Two-Dimensional Pattern Description and Recognition Via Curvature Points," Stanford Linear Acceleration Cent. (USAEC), Stanford University, Stanford, Calif., Tech. Rep. SLAC-70, December 1966.

14. C. T. Zahn, "A Formal Description for Two-Dimensional Patterns," Proc. Int. Conf. Art. Int., pp. 621-628, May 1969.

15. R. S. Leedley, Use of Computers in Biology and Medicine, New York: McGraw-Hill, p. 340, 1965.

16. U. Ramer, "An Iterative Procedure for the Polygonal Approximation of Plane Curves," Comput. Graph. Imag. Proc. 1, pp. 244-256, 1972.

IMAGE CONTOUR REPRESENTATION AND FOURIER DESCRIPTORS

In [8], Zahn discusses the representation of a two-dimensional binary image with a boundary described by a simple closed contour. Since the image is assumed to be binary, it may be defined completely in terms of its boundary. A contour function $\theta(\ell)$ is defined as shown in Figure 5.1. An arbitrary point, S, is chosen as a starting point on the contour Γ. Moving clockwise along the contour, the angle θ between the tangential directions at S and at the current point is associated with the contour length ℓ. For a contour of length L, this association produces a well defined function $\theta(\ell)$ with domain [0, L] for any given starting point S.

As pointed out by Zahn, $\theta(\ell)$ will be invariant under translations and rotations for a fixed starting point. Furthermore, if we introduce the normalized function

$$\phi(t) = \theta\left(\frac{tL}{2\pi}\right) \qquad 0 \leq t \leq 2\pi \quad . \tag{6}$$

We find that $\phi(t)$ has the additional property of being invariant under contour size changes. It is worth noting at this point that in [8] a method is given for reconstruction of the original contour from (6) up to translation, rotation, and size transformations. Since the essential structure of an image will be preserved under these transformations, this result implies that this structural information is completely contained in the function $\phi(t)$.

Since the domain of $\phi(t)$ is given to be $[0, 2\pi]$, we may extend $\phi(t)$ periodically with period 2π and thus represent $\phi(t)$ with the Fourier expansion

$$\phi(t) = \mu_o + \sum_{k=1}^{\infty} A_k \cos(kt - \alpha_k) \quad , \tag{7}$$

with amplitude and phase parameters, A_k and α_k. These parameters are termed as the Fourier descriptors in [8]. As Zahn points out, the phase parameters only are a function of the starting point. Thus, the A_k's depend only upon the shape of the contour, being independent of translation, rotation, and size transformations and the starting point S. As will be mentioned subsequently, this fact suggests a natural similarity measure for the class of images under discussion. In the next section, we leave the Fourier descriptor approach and discuss the description of the contour in terms of impulses.

PICTURE DESCRIPTION USING PULSE SEQUENCE REPRESENTATION

A common technique in image analysis is to represent smooth countours like those discussed in the previous section by a polygonal approximation. In particular, the digitized version of the smooth contour, as is most frequently treated in image processing, is in fact a polygon. Therefore, it is quite reasonable to restrict the class of images under discussion to those with polygonal boundaries as in Figure 5.2a. With this assumption, we find that $\phi(t)$ becomes a constant along the edges of the polygon, with jump discontinuities at the vertices as shown in Figure 5.2b. In general, $\phi(t)$ may be expressed as

$$\phi(t) = \sum_{i=1}^{N} a_i \, U\left(\frac{tL}{2\pi} - \ell_i\right) \quad , \tag{8}$$

where

a_i is the angular change corresponding to the vertex at distance ℓ_i from the starting S, and

$U(t)$ is the unit step function.

Using Equation (8), we may now derive the contour representation to be discussed in the remaining sections of this paper.

Taking the derivative of $\phi(t)$ in Equation (8), we find that $d\phi(t)/dt$ takes the form

$$\frac{d\phi(t)}{dt} = \sum_{i=1}^{N} a_i \, \delta(t - t_i) \quad , \tag{9}$$

where

$\delta(t)$ denotes the unit impulse function, and

$$t_i = \frac{2\pi}{L} \ell_i \quad .$$

Thus, the polygonal contour may be uniquely defined in terms of a sequence of pulses as illustrated in Figure 5.2c. It may be further pointed out that the concave and convex angles of the polygon are related to the pulse amplitude sign. The concave angles correspond to positive amplitudes, while the convex angles are negative.

Intuitively, one associates the complexity of a figure to the concave angles in the figure. This suggests the following complexity measure:

$$C = \int_0^{2\pi} \left| \frac{d\phi(t)}{dt} \right| dt - 2\pi \quad . \tag{10}$$

For a polygon, Equation (10) may be written in terms of the concave angles as

$$C = 2 \sum \text{(concave angles)} \quad . \tag{11}$$

Thus, Equation (10) is a measure of the concavity in the contour.

Clearly, Equation (10) is a useful measure for distinguishing images of different complexity, such as a rectangle from a cross. On the other hand, for objects of similar complexity, such as a rectangle and a triangle, the complexity measure does not contain sufficient discriminating information and so we must extract more features from the pulse sequence. In the worst case, the pulse sequence itself may be required for classification of images. Thus, the feature reduction problem may be viewed at successive simplifying operations on the pulse sequence where the limiting case will be one feature, such as the complexity measure in Equation (10).

Figure 5.3 illustrates some possible simplification. In Figure 5.3b, all adjacent convex angles are combined, thus preserving the convex/concave relation of the image. In Figure 5.3c, the convex angles of Figure 5.3b are all assumed to be $-\pi$. Here, the concave angles are retained along with their ordering relation with the convex angles. The final simplification shown in Figure 5.3d combines all adjacent concave angles in Figure 5.3c. One final simplification to Figure 5.3d would be to add up all concave angles which is equivalent to the complexity measure of Equation (10).

We would like to conclude this section with a discussion of the representation of the skeleton of an image in terms of a pulse sequence. In this case, the skeleton is traced out as shown in Figure 5.4, with end points in the skeleton represented by convex angles of π radians. A skeleton composed of line segments may be systematically constructed from smaller skeletons. The process of joining two skeletons and the corresponding merging of their respective pulse sequences is illustrated in Figure 5.5. The new pulse sequence is formed by shifting the pulse sequence of skeleton B such that its starting point is at S'. The end points of this sequence are then replaced by concave angles α and $\pi - \alpha$ in the case of Figure 5.5c or α and $-\alpha$ in the case of Figure 5.5d. Finally, this pulse sequence is inserted in the pulse sequence for skeleton A at the point S. In the case of Figure 5.5d, the $-\pi$ pulse at b is replaced by the pulse sequence of skeleton B.

Using this basic construction method, a complex skeleton may be constructed along with its pulse representation from simple line segments. In addition, a pulse sequence will represent a skeleton only if it may be constructed in this manner. Thus, an algorithm based upon this construction can be used to derive a skeleton pulse sequence from the original contour pulse sequence.

In the next section we consider the corruption of the pulse functions, discussed here, by image noise in the contour.

SIMILARITY MEASURE

As previously mentioned, Zahn and Roskies propose a similarity measure between two contours, γ and γ', in terms of the Fourier description of $\phi(t)$. This measure is defined as

$$S_A[\gamma, \gamma'] = \sum_{k=1}^{\infty} |A_k - A_k'| \quad , \tag{12}$$

where the A_k's are the amplitude coefficients in Equation (7). For our purposes, we would like to consider the expression

$$\dot{\phi}(t) = \mu_o + \sum_{k=1}^{\infty} B_k \cos(kt - B_k) \quad , \tag{13}$$

and define the similarity

$$S_B^{(k)}[\gamma, \gamma'] = \sum_{k=1}^{k} |B_k - B_k'| \quad , \tag{14}$$

where k represents the cutoff frequency. As in the case of the A_k's, the B_k's are independent of rotation, translation, contour length, and starting point. Therefore, Equation (14) provides a distance measure between γ and γ' which depends only upon shape. For a $\dot{\phi}(t)$ given by Equation (9), B_k may be written as

$$B_k = \frac{1}{\pi} \left| \sum_{i=1}^{n} a_i \, e^{-jkt_i} \right| \quad , \tag{15}$$

where $|\cdot|$ denotes the modulus of the complex quantity.

The similarity measure in Equation (14) clearly has discriminating ability as is shown in several explicit examples to follow in this and subsequent sections. The limitations of this discrimination are also shown to be a function of the parameter k. This fact is demonstrated to be advantageous in the next section for classifying contours to various shapes in the presence of noise. The examples to follow in this section reveal that low values of k produce significant similarity distances between images such as triangles, squares, and rectangles. While on the other hand in the next section, we see that noise in an image can be largely eliminated from the similarity measure if a low value for k is chosen.

We conclude this section with explicit examples of calculations for Equation (14). For these examples, we consider the similarity between the square, rectangle, triangle, and cross shown in Figure 5.6. Table 5.1 shows the theoretical similarity measure for various values of k.

NOISE CONSIDERATIONS IN DIGITIZED PICTURES

In this section, we discuss the types and effects of noise which occur in real digitized pictures. Consider a binary picture which is digitized into a matrix of (0, 1) pixel values. As we see by the example in Figure 5.7, the contour is composed of line segments of fundamental length B. For simplification of the discussion, we are assuming the diagonal segments to have the same length as the vertical and horizontal segments. Because of the pixel structure of the image, the angle at a vertex is considered to take on one of seven possible values ($\pi/2$, $\pi/4$, 0, $-\pi/4$, $-\pi/2$, $-3\pi/4$). Therefore, the contour function $d\phi(t)/dt$ for a digitized picture of this type will be a sequence of pulses $B \cdot \frac{2\pi}{L}$ apart with each pulse taking on one of seven values. Thus, the general pulse function will have the form

$$\dot{\phi}(t) = \sum_{i=1}^{n} a_i \, U(t - i \cdot \frac{2\pi}{n}) \quad , \tag{16}$$

where n is the number of incremented segments in the contour. For typical digitized pictures, the image size is large compared to the pixel size, for example, n on the order of 100.

Two types of noise appear in a digitized contour of this form. The first type is what we call digitization noise. This noise will occur whenever a smooth contour is digitized into fixed length line segments with a finite number of possible vertex angles. An example of this digitization noise is given in Figure 5.8a. In this example, we see that the sides of the digitized contour oscillate about the sides of the triangle. The second type of noise

TABLE 5.1. SOME SIMILARITY MEASURES

	SQUARE	RECTANGLE	CROSS	TRIANGLE
K = 4				
Square	0	2.0	1.95	4.12
Rectangle		0	2.13	4.84
Cross			0	2.71
Triangle				0
K = 6				
Square	0	4.0	6.84	6.9
Rectangle		0	5.02	5.11
Cross			0	5.33
Triangle				0
K = 8				
Square	0	5.0	8.28	8.39
Rectangle		0	7.41	6.10
Cross			0	8.25
Triangle				0

is image noise. In this case, the noise is inherent in the detected image by a noisy background in the picture. This noisy background will cause the detected boundary contour to deviate from the actual contour. An example of this is given in Figure 5.8b, where the side of the square has been obscured by noise.

Figure 5.9 shows two fundamental noise elements with their corresponding pulse components. An image contour which is distorted by one of these noise elements may be represented by

$$\dot{\phi}_{I+N}(t) = \dot{\phi}_I(t) + \dot{\phi}_N(t) \quad , \tag{17}$$

where $\dot{\phi}_I(t)$ is the undisturbed contour, while $\dot{\phi}_N(t)$ represents the noise. For the noise element in Figure 5.9a (type A noise),

$$\dot{\phi}_N(t) = \frac{\pi}{4}\,\delta\left(t - \frac{m}{n}\,2\pi\right) - \frac{\pi}{2}\,\delta\left(t - \frac{(m+1)}{n}\,2\pi\right) + \frac{\pi}{4}\,\delta\left(t - \frac{(m+2)}{n}\,2\pi\right) \quad , \tag{18}$$

where again n is the number of pixel segments in the image contour. The noise element in Figure 5.9b (type B noise) will be equivalent to Equation (18) with a sign change. We may now proceed to calculate an upper bound on the similarity distance between the original and noisy contours.

To derive a bound for Equation (12), where γ and γ' are the original and noisy contours, respectively, let (B_k, α_k), (B_k', α_k'), and $(B_k^{(N)}, \alpha_k^{(N)})$ represent the Fourier parameters in Equation (13) for $\dot{\phi}_I(t)$, $\dot{\phi}_{I+N}(t)$, and $\dot{\phi}_N(t)$. By Equations (13) and (17), we have

$$B_k' \cos(kt - \alpha_k') = B_k \cos(kt - \alpha_k + B_k^{(N)} \cos(kt - \alpha_k^{(N)}) \quad . \tag{19}$$

Using the complex exponential, Equation (19) may be written as

$$\mathrm{Re}[B_k' e^{j(kt - \alpha_k')}] = \mathrm{Re}[B_k e^{j(kt - \alpha_k)}] + \mathrm{Re}[B_k^{(N)} e^{j(kt - \alpha_k^{(N)})}] \quad . \tag{20}$$

Since Equation (20) holds for all t, Equation (20) reduces to

$$B_k' e^{-j\alpha_k'} = B_k e^{-j\alpha_k} + B_k^{(N)} e^{-j\alpha_k^{(N)}} \quad . \tag{21}$$

Now, from Equation (21) and the fact that B_k, B_k', and $B_k^{(N)}$ are defined to be positive, one can derive the following inequality:

$$|B_k' - B_k| = ||B_k' e^{-j\alpha_k'}| - |B_k e^{-j\alpha_k}|| \leq |B_k' e^{-j\alpha_k'} - B_k e^{-j\alpha_k}| = B_k^{(N)} \quad . \quad (22)$$

Finally, we now have from Equations (14) and (22),

$$S_B^{(k)}[\gamma, \gamma'] \leq \sum_{k=1}^{k} B_k^{(N)} \quad . \quad (23)$$

Thus, the similarity distance between a noisy contour and the noise free contour is bounded by the expression on the right-hand side of Equation (23), which may be calculated from the elemental noises of Figure 5.9. Furthermore, typical noises may be modeled by adding up several fundamental noise elements which are shifted along the contour with respect to each other, that is, for different values of m in Equation (18). For example, in Figure 5.8a, the digitization noise may be modeled by a sequence of alternating types of noise elements. Similarly, for Figure 5.8b, the noise may be considered as the sum of two type A noises and four type B noises with the appropriate positioning along the contour. Using this addition of fundamental noise elements to characterize noise, we can derive a general bound analogous to Equation (23):

$$S_B^{(k)}[\gamma, \gamma'] \leq \sum_{\xi=1}^{N_\xi} \left(\sum_{k=1}^{k} B_k^{(\xi)} \right) \quad , \quad (24)$$

where N_ξ is the number of fundamental elements in the noise.

Using Equation (15) we can calculate the right-hand side of Equation (24) for the noises in Figure 5.9 to be

$$\sum_{k=1}^{k} B_k^{(N)} = \frac{a}{2\pi} \sum_{k=1}^{k} |1 - 2 e^{-j\frac{2\pi k}{n}} + e^{-j\frac{4\pi k}{n}}|$$

$$= \frac{2a}{\pi} \sum_{k=1}^{k} |\sin \frac{\pi k}{n}| \quad . \quad (25)$$

Substituting Equation (25) into Equation (24),

$$S_B^{(k)}[\gamma, \gamma'] \leq \left(\sum_{\xi=1}^{N_\xi} a_\xi \right) \frac{2}{\pi} \sum_{k=1}^{k} |\sin \frac{\pi k}{n}| \quad , \quad (26)$$

gives a general upper bound for the noise. By noting the sine terms in Equations (25) and (26), we see that the low order terms in the summation will have negligible effect, that is, for $\pi k/n$ small. Therefore, an upper bound in choosing k will depend upon this sine function envelope. On the other hand, the lower bound for k will be determined by the ability of the similarity measure to distinguish different images as discussed in the previous section. In the remainder of this section, we give some explicit examples using the similarity measure to identify noisy contours.

For these examples, we have simulated digitization and image noise for the four basic shapes in Figure 5.6. These are shown in Figure 5.10. In the final section, we discuss a technique for filtering the noise out of $\dot{\phi}(t)$ and thus restoring a noise contour.

CONTOUR RESTORATION

In our final section, we discuss a technique for taking advantage of the high frequency characteristics of contour noise to extract a noiseless contour from a noisy one. In the previous section, we found that the noise envelope is given by $\sin(kt/n)$. Therefore, for small values of k, the noise contribution will be minimal. This suggests that if the signal $\dot{\phi}(t)$ is passed through a low pass filter, the noise component can be all but eliminated and thus the characteristics of the filtered signal will reflect the original contour. If the noise in the contour is small (in terms of deviation from the original signal), the window width of the filter may be chosen to filter out the noise while retaining the basic structure of the contour. To reconstruct the filter, we must next use a sharpening filter to extract the position and angle of the original vertices of the contour. Essentially, this process is one of finding the best pulse sequence represented by the filtered $\dot{\phi}(t)$. The system diagram as shown in Figure 5.11 thus consists of a smoothing filter followed by a sharpening filter. By choosing the right filter parameters, we will show that only the original contour characteristics will be retained.

For the specific examples to follow, we have chosen the filtering process to be

$$\dot{\phi}_{filt}(t) = \dot{\phi}_e(t) * g(t) \quad , \tag{27}$$

where $\dot{\phi}_e(t)$ is the periodic extension of $\dot{\phi}_{I+N}(t)$ and

$$g(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-t^2/2\sigma^2} \quad . \tag{28}$$

Equation (28) has been chosen for g(t) primarily for its smoothness and unimodal structure. The unimodal structure has been found to greatly facilitate the sharpening operation. The sharpening filter used consists of locating the major positive peaks and negative valleys. These points are then identified as concave and convex angles, respectively. The values are then determined by integrating under each peak and valley and multiplying by a proportionality constant so that the angles add up to -2π since $\int_0^{2\pi} \dot{\phi}(t)dt = -2\pi$ as previously mentioned. Figure 5.12 shows the filtered pulse functions and their sharpened versions corresponding to the noisy contours in Figure 5.10. Figure 5.13 demonstrates the application of this filtering and sharpening technique to a typical aircraft contour.

CONCLUSIONS

In this paper we have discussed a method for representing binary images with simple closed boundaries. In particular, for polygonal boundaries, the representation we use is a one-dimensional function formed by a sequence of impulse functions. Using this representation, we have discussed the feature extraction problem and proposed a complexity measure which is shown to be related to the concavity of the contour. Next, we introduce a similarity measure which is analogous to the one Zahn proposes, but is in terms of $\dot{\phi}(t)$. Using this measure, we have discussed the effect of noise imposed upon the contour. It was shown that this noise shows a high frequency characteristic and thus can be filtered out. In addition, using only the low frequency terms for the similarity measure, we can use this measure for identifying noisy contours. Finally, we have also demonstrated a method for extracting the original contour from noise. This method consists of a smoothing filter followed by a sharpening filter. This method is demonstrated by a few explicit examples.

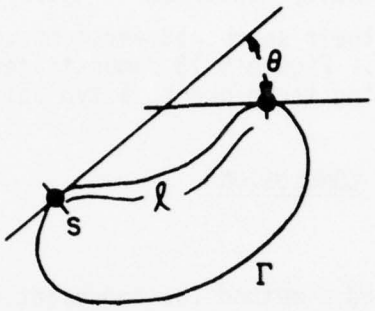
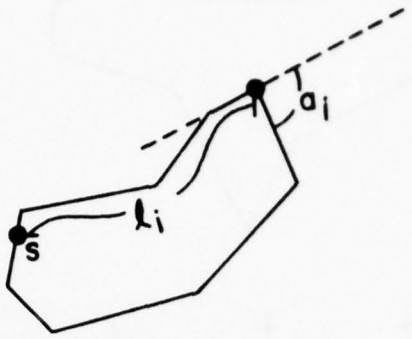
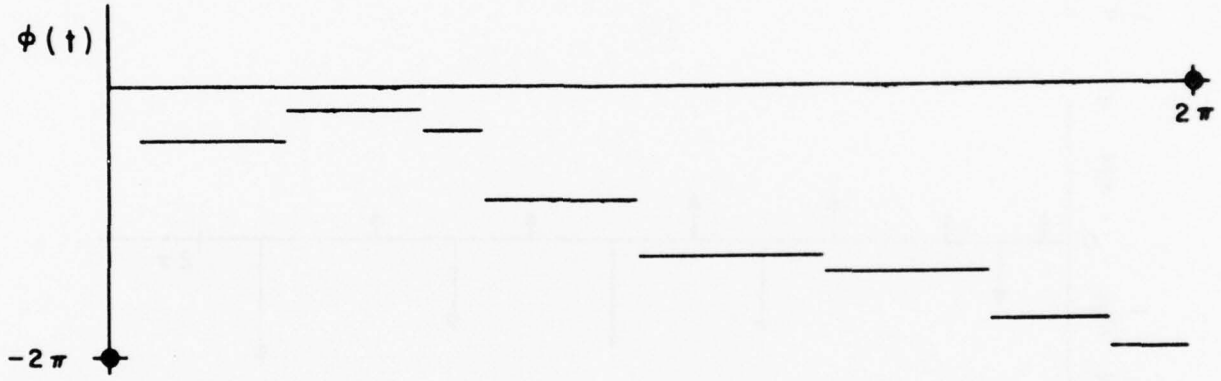


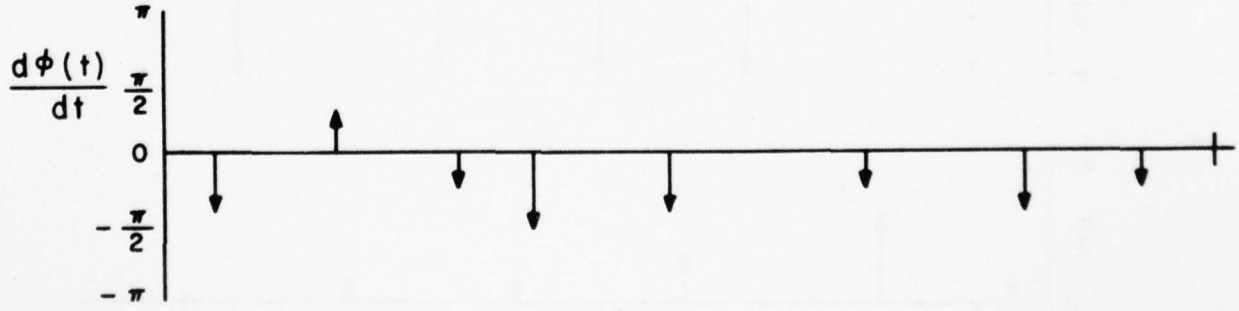
Figure 5.1. Contour Representation



(a)



(b)



(c)

Figure 5.2. Polygonal Representation of Figure 5.1

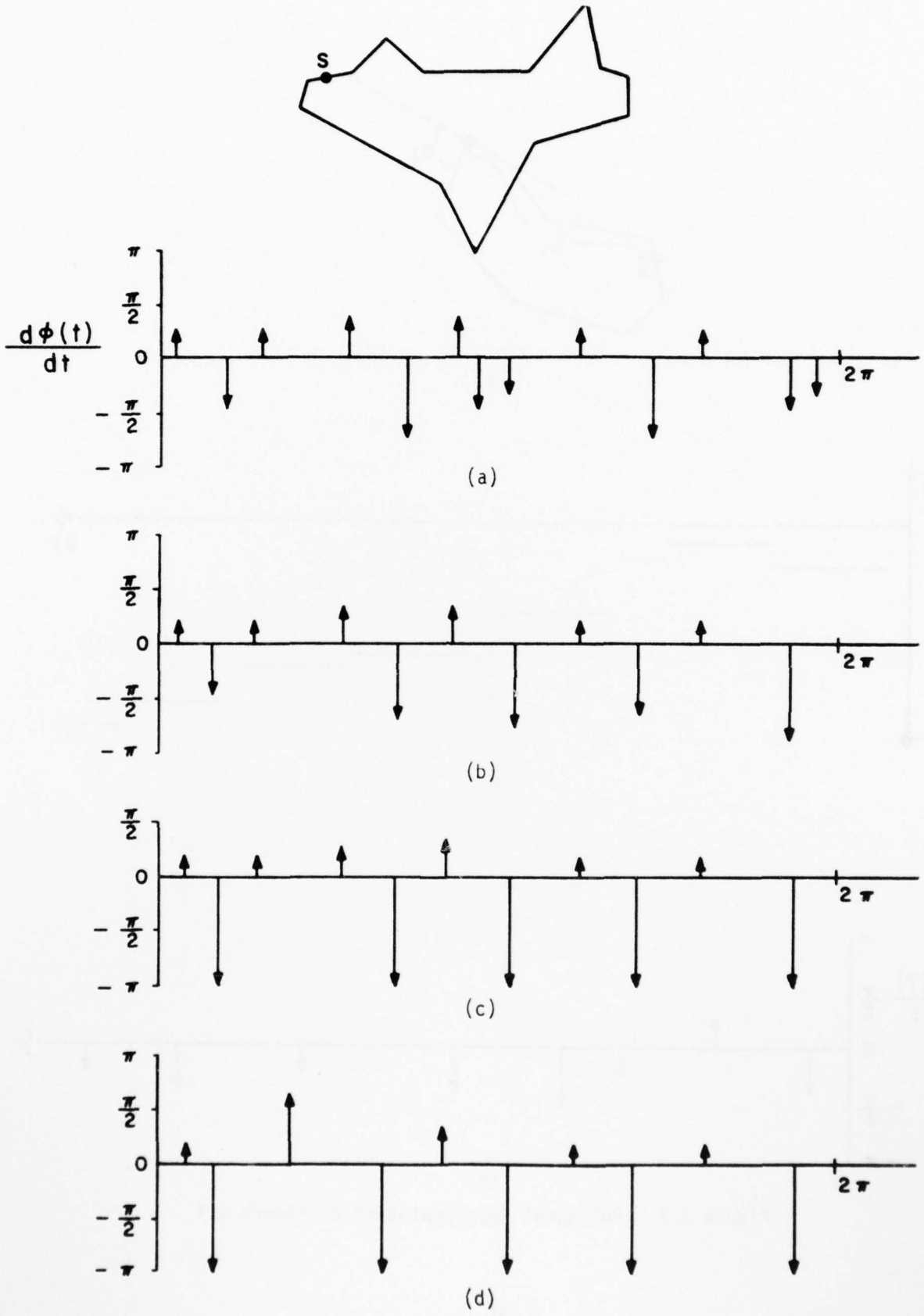


Figure 5.3. Successive Simplifications of a Pulse Sequence Representation

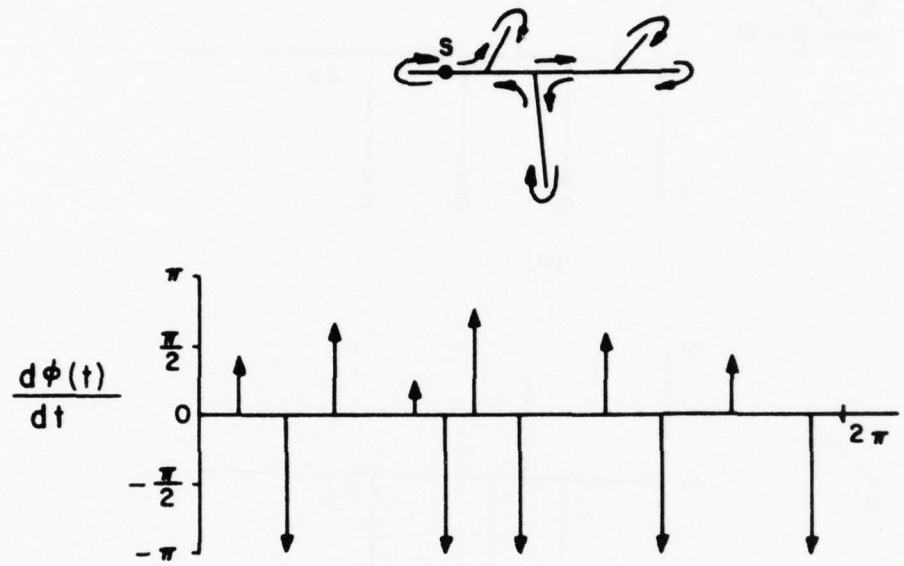
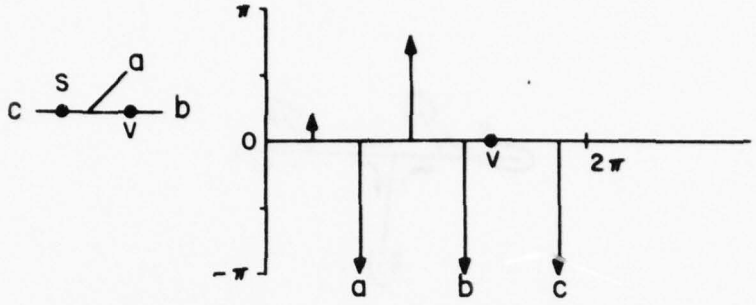
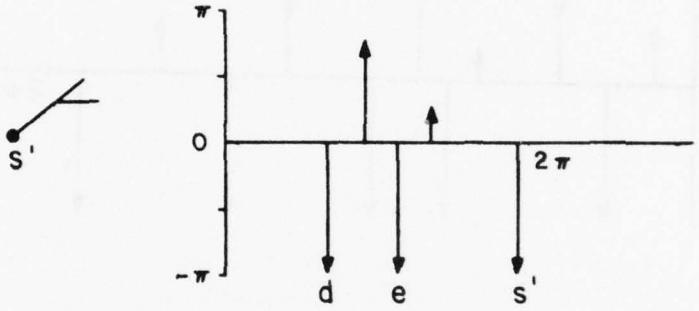


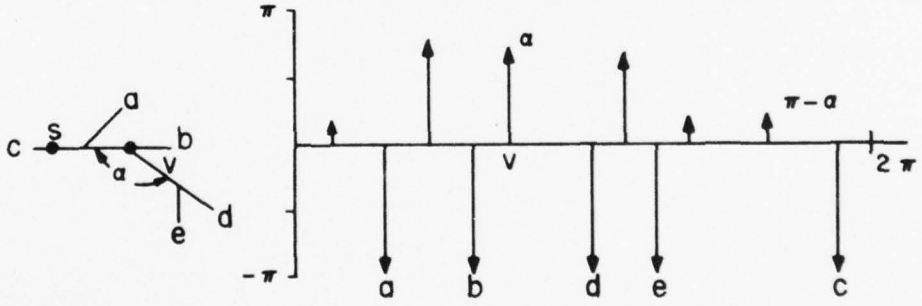
Figure 5.4. Pulse Representation of a Skeleton



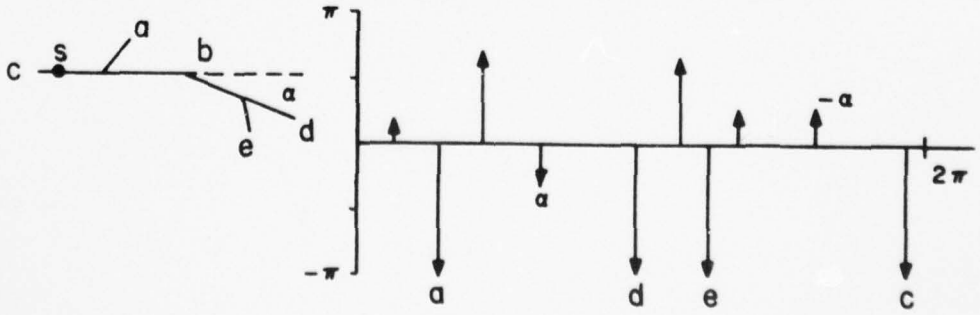
(a)



(b)



(c)



(d)

Figure 5.5. Skeleton Merging

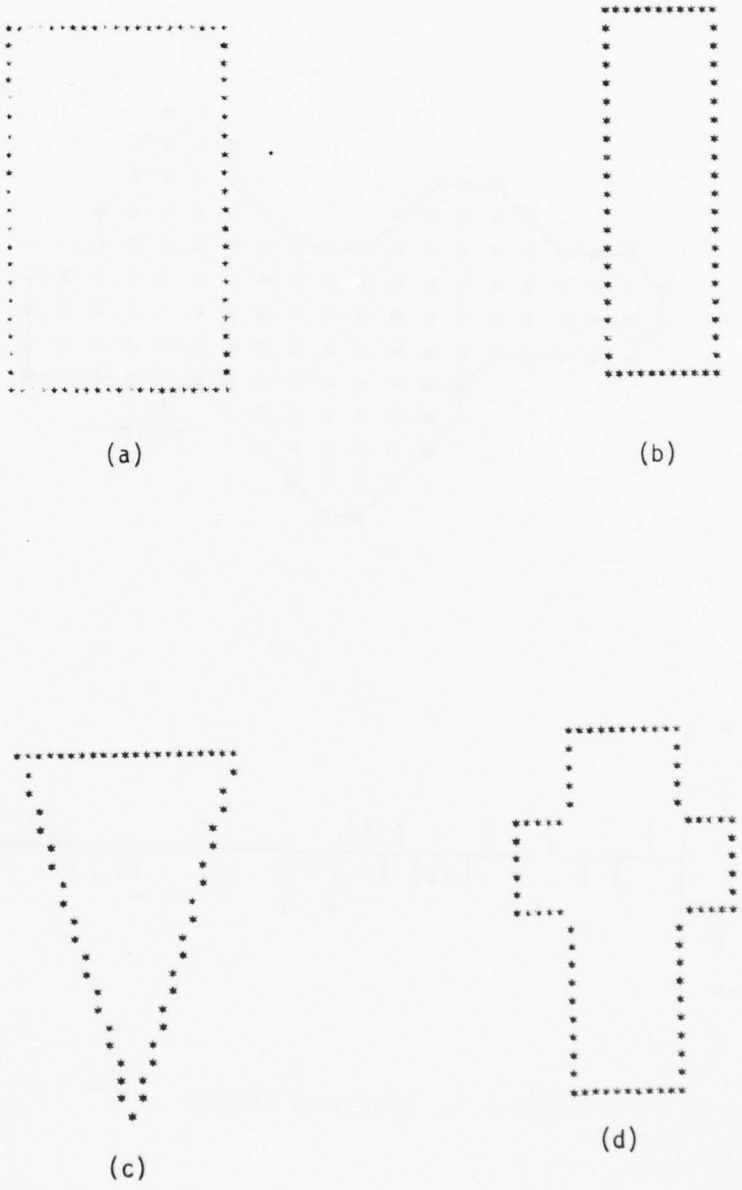


Figure 5.6. Basic Image Shapes

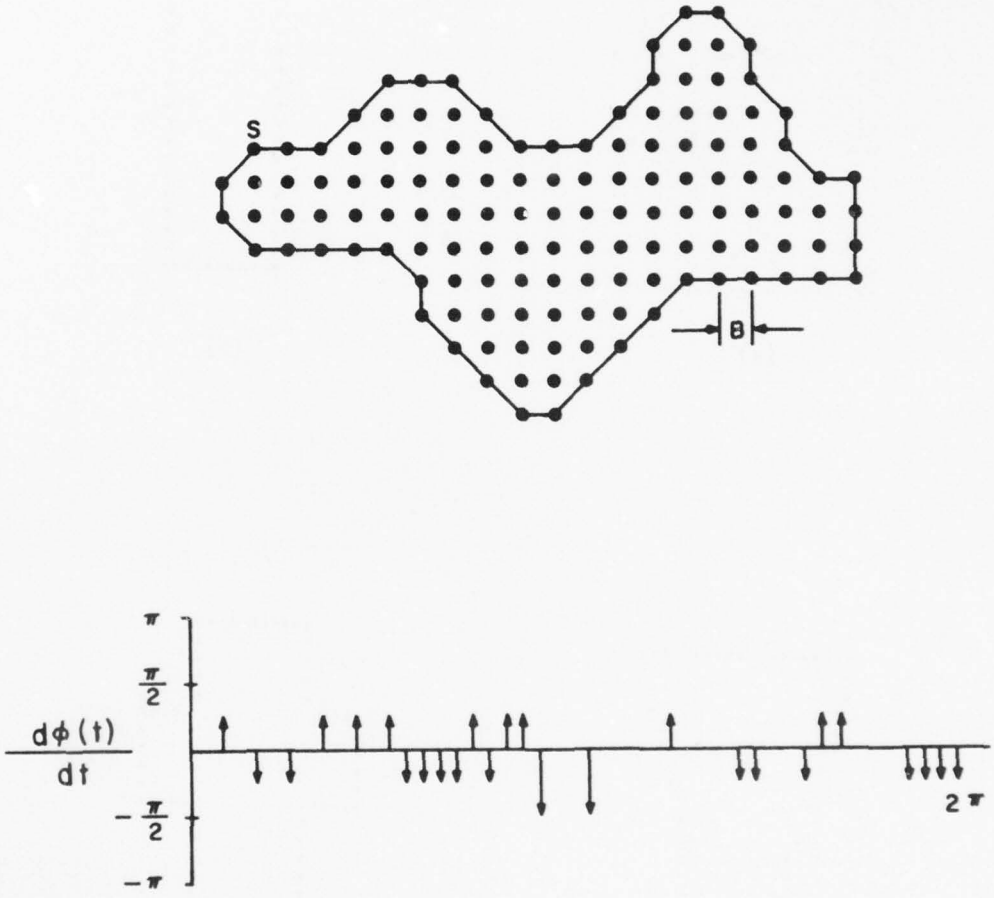
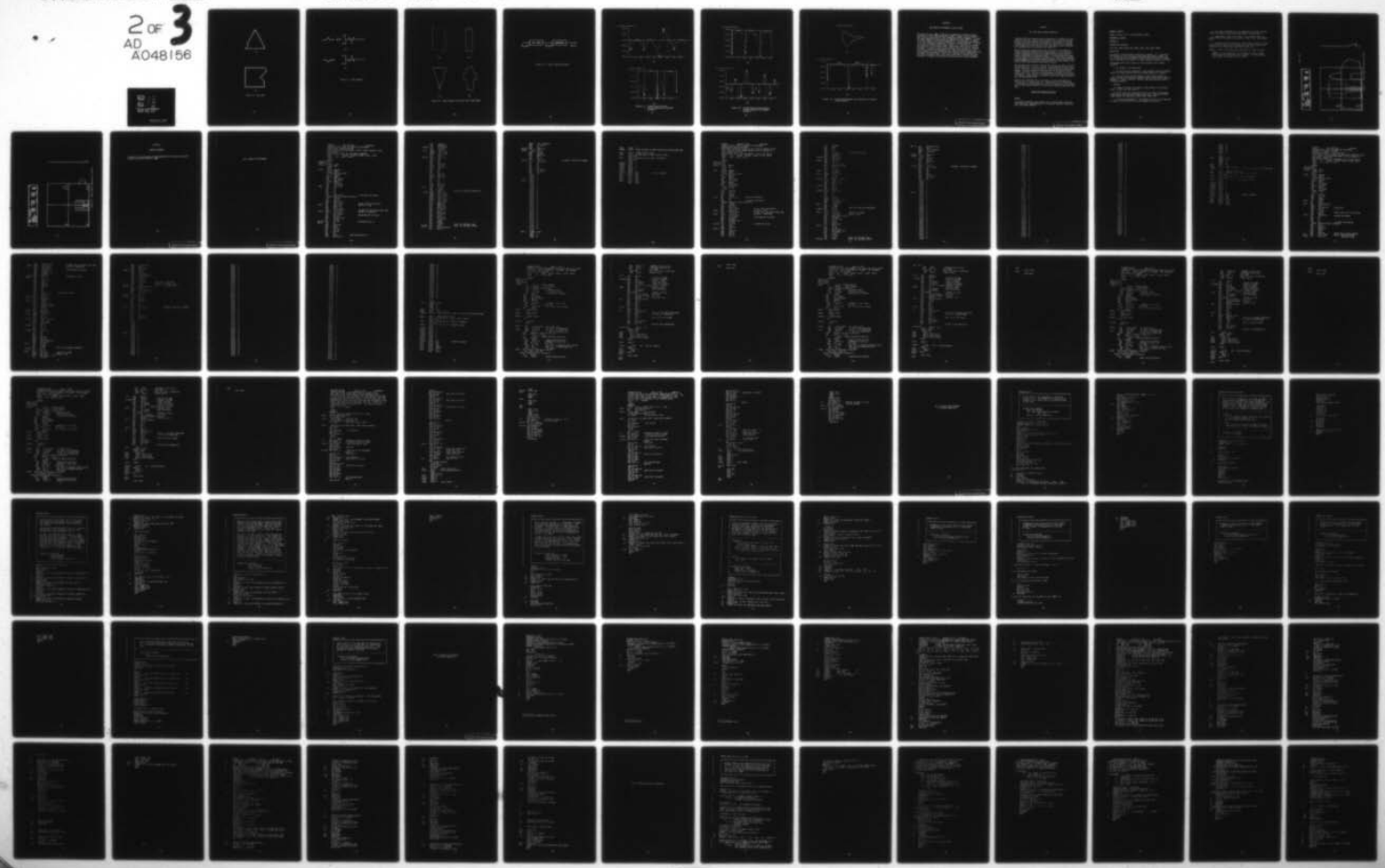


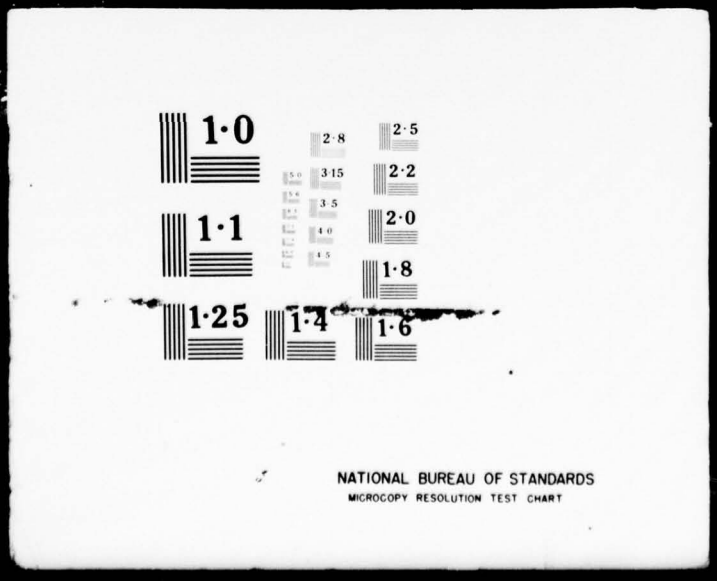
Figure 5.7. Digitized Picture

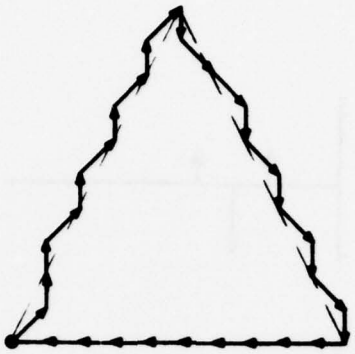
AD-A048 156 WHITE SANDS MISSILE RANGE N MEX INSTRUMENTATION DIRE--ETC F/G 17/8
SEGMENTATION AND STRUCTURE ANALYSIS FOR REAL-TIME VIDEO TARGET --ETC(U)
OCT 77 K FUKUNAGA , A L GILBERT, M K GILES

UNCLASSIFIED STEWS-ID-77-1 NL

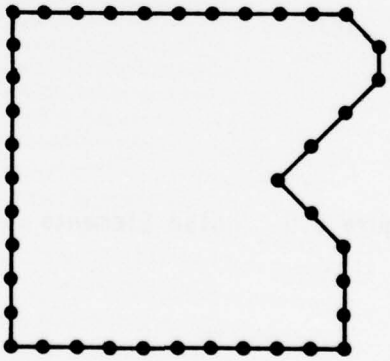
2 OF 3
AD
A048156





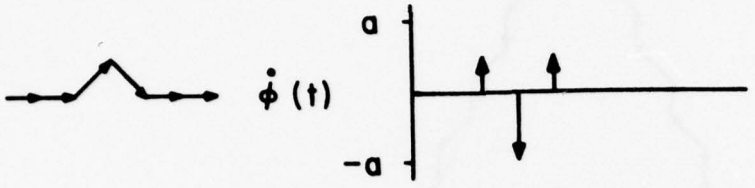


(a)

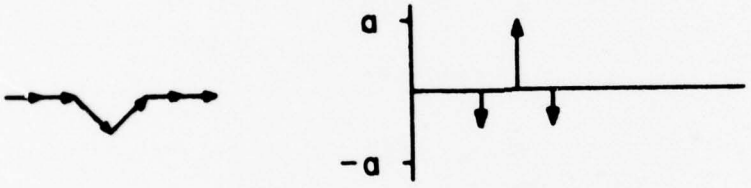


(b)

Figure 5.8. Noise Types



(a)



(b)

Figure 5.9. Noise Elements

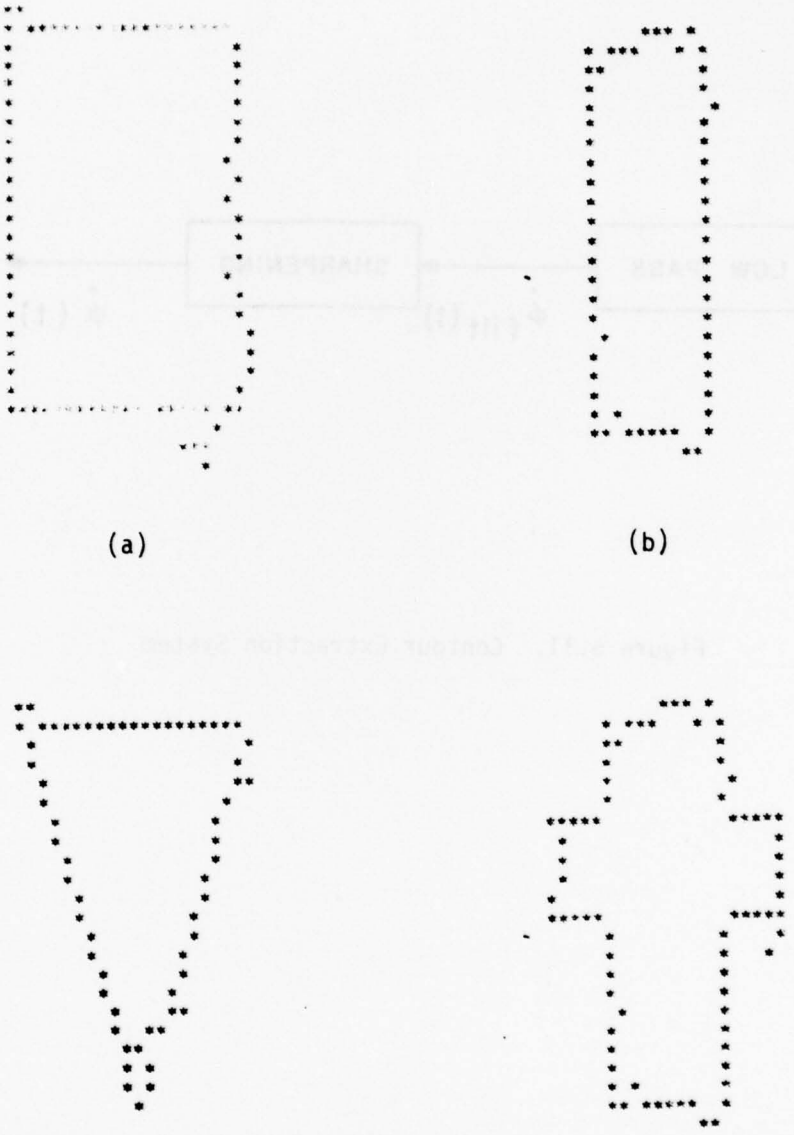


Figure 5.10. Noisy Contours of the Four Basic Image Shapes

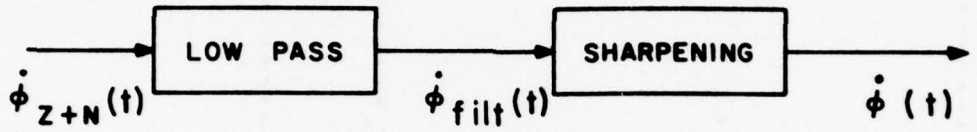
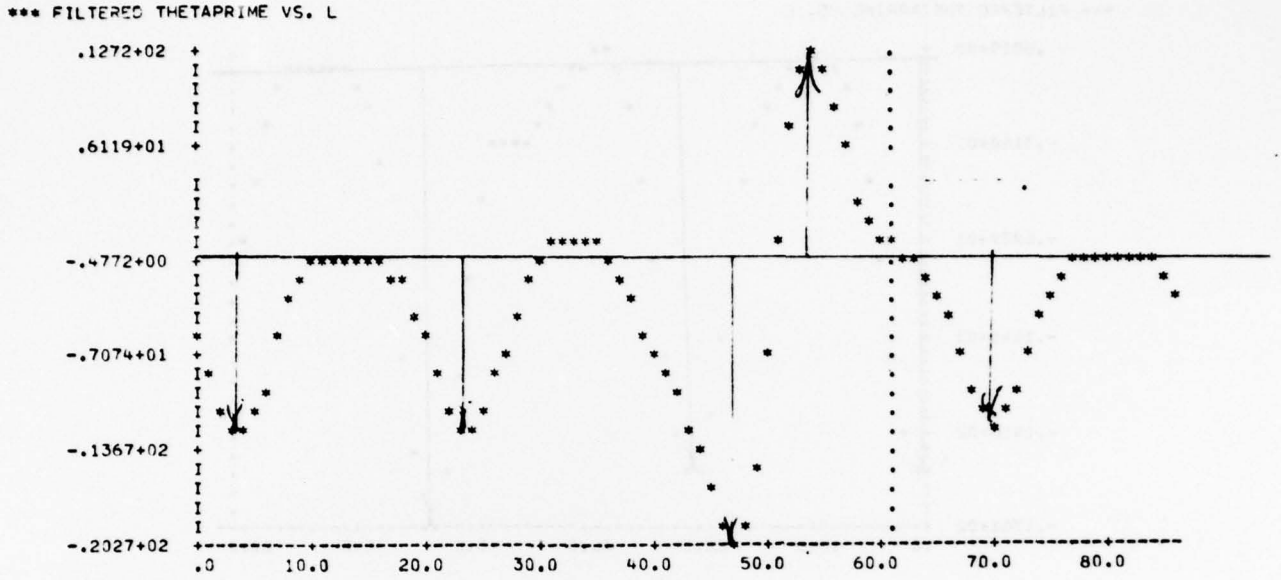
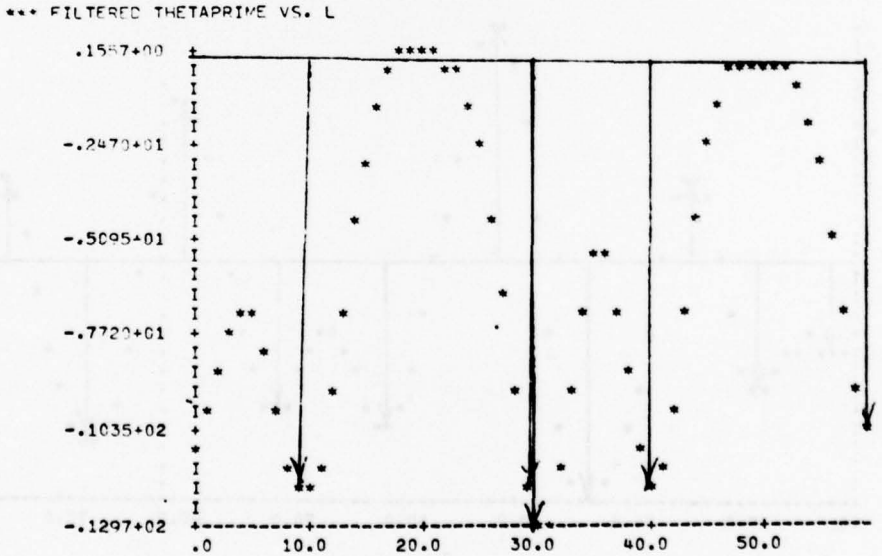


Figure 5.11. Contour Extraction System

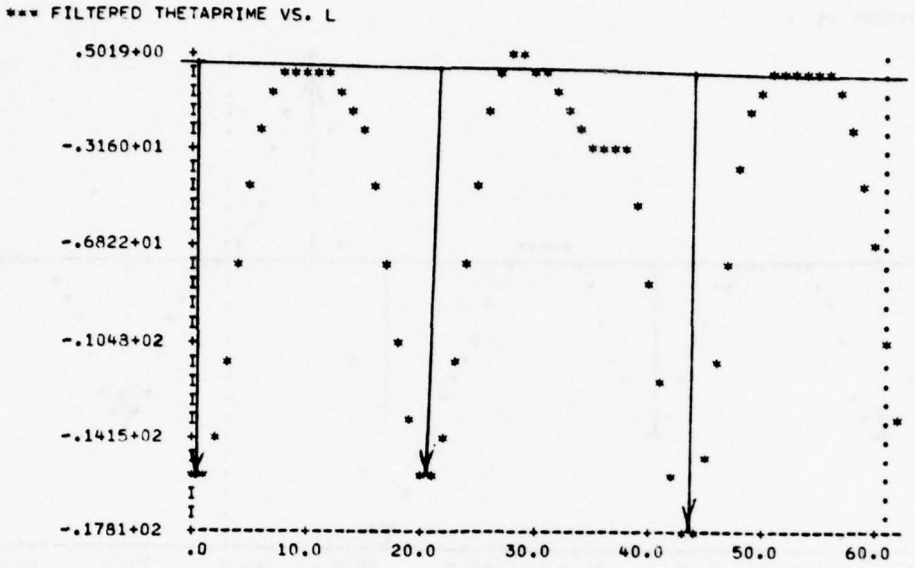


(a)

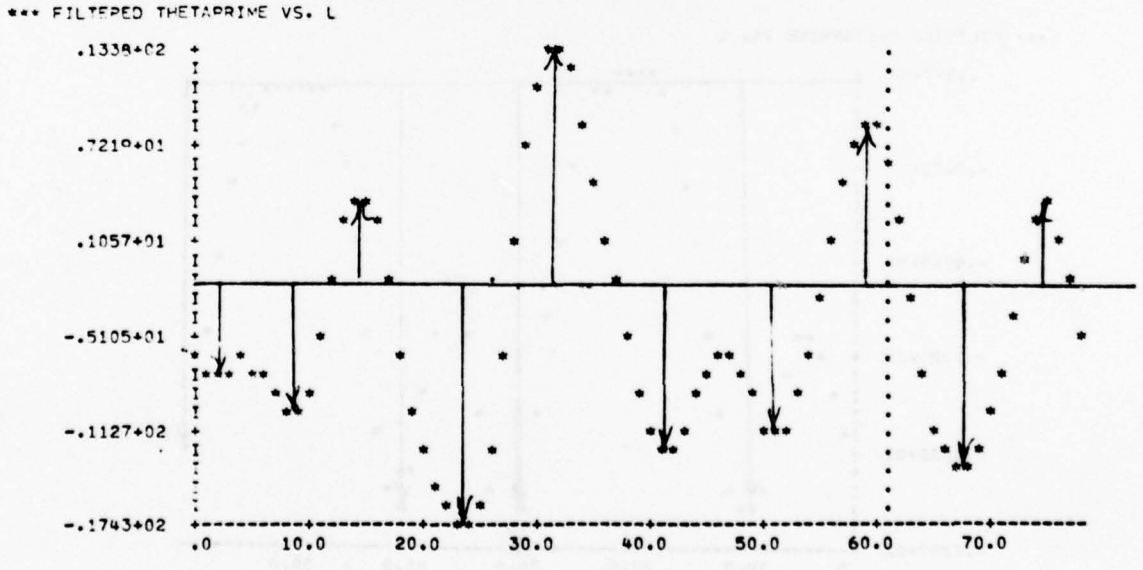


(b)

Figure 5.12. Filtered Pulse Functions (Arrows represent the sharpened versions)

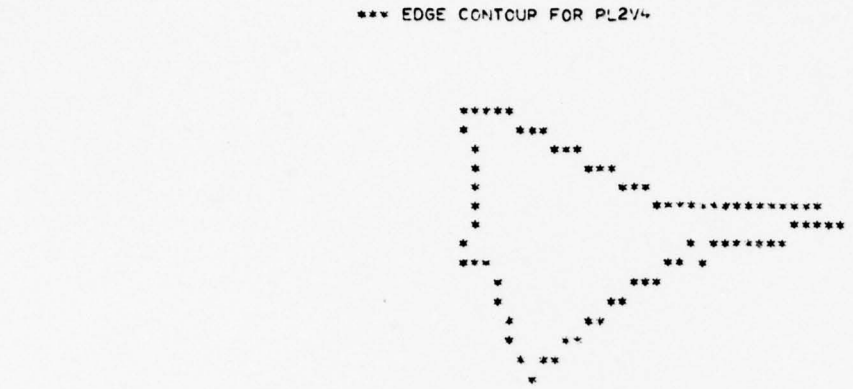


(c)

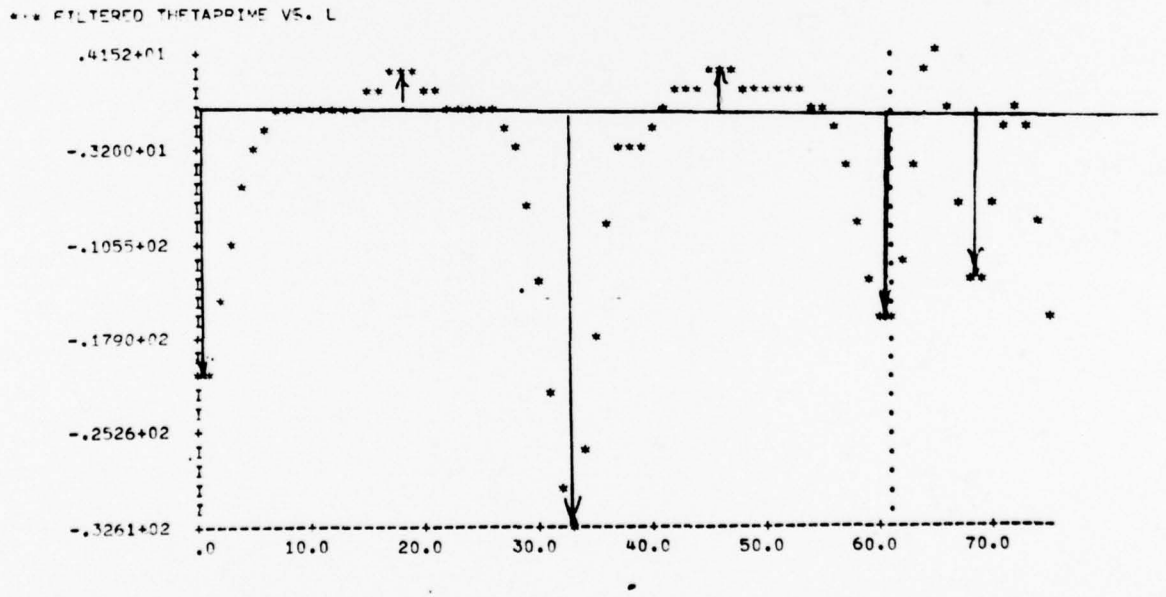


(d)

Figure 5.12. Filtered Pulse Functions (Cont'd) (Arrows represent the sharpened versions)



(a)



(b)

Figure 5.13. Filtered and Sharpened Pulse Function of a Typical Aircraft Contour

SECTION 6

RELATIONSHIP OF RESEARCH TO THAT OF NMSU

One purpose of this summer's effort was to compare our proposed tracking techniques to those being implemented by NMSU. However, at this stage of data collection and analysis, such a comparison would be premature. It seems the image decomposition and projection computation used by NMSU is indeed relatively simple for real-time implementation. The question to be answered is "is this computation sufficient for typical tracking problems?" It is apparent that as hardware capabilities increase, it is desirable to increase the complexity of the system to make it more robust under various tracking situations. We plan to continue this research and to continue to cooperate with NMSU in establishing and analyzing a video tracking data base. Our special emphasis will be segmentation and structure analysis algorithms--their advantages and shortcomings. We will also investigate the combination of the various techniques into a high level system that can choose the proper techniques in sequence and interpret results to control the next requests.

SECTION 7

REAL-TIME VIDEO TRACKER SIMULATION

A general effort was directed toward coordination of the NMSU real-time video (RTV) tracker research and the Purdue picture processing research. Specific effort was directed toward obtaining video tapes of a variety of WSMR missile tests and their digitization, coordination of meetings with the NMSU RTV tracker research staff, and general system integration and troubleshooting of the digitzer and video disk.

Specific effort was directed toward the installation of the NMSU RTV tracker simulation program on the Tektronix system. Since the original simulation program was written for an HPMX21/Tektronix 4004 graphics system, transportability problems existed. Dr. Raul Machuca was responsible for rewriting the simulation software. Dr. Javin Taylor supported the development of the top-down program structure and performed the final program debugging and the rewriting of the NMSU simulation graphics software for use on the Tektronix 4014 graphics system.

Two subroutines were written to enhance the initial user setup of the RTV tracker simulation. The first provides a three-dimensional view of the missile trajectory so that the user can visualize how the tracker should respond. The second subroutine provides plots of the dynamic tracker azimuth, elevation, rotation, and zoom response for perfect missile trajectory data. Thus, the user can determine how tracking errors should be attributed to tracker dynamics limitations rather than image recognition and structure decomposition.

Analysis was initiated, and is continuing, to shorten the processing time for the RTV tracker simulation. Analysis was also initiated, and is continuing, to determine how real digitized video data can be inserted into the RTV tracker simulation in place of the simulated digitized video data.

SIMULATION PROGRAM DESCRIPTION

PURPOSE

To simulate a complete video tracking loop including target trajectory, video simulation, image decomposition, structural tracking, tracker control, and tracker dynamics.

HARDWARE REQUIRED

Graphics terminal, PDP 11/35 minicomputer system.

PROGRAMMING LANGUAGE

FORTRAN IV.

FORTRAN FILES REQUIRED

RTVS, RTV, CONTR, DCOMP, GEN, TKRDY, TRJEC, PCM, STRPC, GRAPHS.

DESCRIPTION

The purpose of the RTV tracker simulation is twofold: (1) to simulate the operation of the RTV tracker system under development at NMSU, and (2) to provide a test bed whereby picture processing and pattern recognition algorithms can be tested in a hybrid simulator/hardware environment.

The program starts with a series of initial questions which require responses:

1. Any changes? Self explanatory.

2. 3D plot of missile trajectory? A yes response causes the graphics terminal to display the path that the simulated target will traverse.

3. Plot of perfect tracker response? A yes response causes the graphics terminal to display the dynamic response of the various tracking components: azimuth, elevation, rotation, and zoom based upon perfect tracking information.

OPTIONS:

(1) Number of frames? The number of video frames to be simulated. Enter 1 to 360, right justified I3.

(2) Which plot, etc? Selection of Az, El, rotation, or zoom plots target versus time and tracker output versus time. Selection of AZERR, ELERR, ROTERR, and ZMERR plots tracker error versus time.

4. Ask every frame Modulo N? The program will halt at the frame number modulo the specified N and ask the sequence of questions.

5. Plot every frame Modulo N? The program will plot the simulated video and tracking at every frame number modulo the specified N.

6. Dump present state of the system? A yes response causes the present state of the system to be stored in a data file on the simulator disk.

7. Load new state for the system? A yes response causes the system state stored in a data file on the simulated disk to be loaded so the simulator can be started from a previously stored state.

Figures 7.1 and 7.2 show video frame simulations at frames 19 and 98.

WARNING: At the present time, do not request the zoom or ZMERR option under plot of perfect tracker response. The zoom response has not been incorporated into this option.

0=NO, 1=YES -

	TARGET	PREDICTED	TRACKER	
TIME = 0.317 SEC.				19
RANGE = 20001.012 FT.				
AZIMUTH	0.0000	0.0000	0.0000	RAD.
ELEVATION	0.0101	0.0100	0.0095	RAD.
ROTATION ANGLE	0.0000	0.0000	0.0000	RAD.
VIEW ANGLE	1.5807			RAD.
ZOOM RATIO		3.64:1	3.64:1	

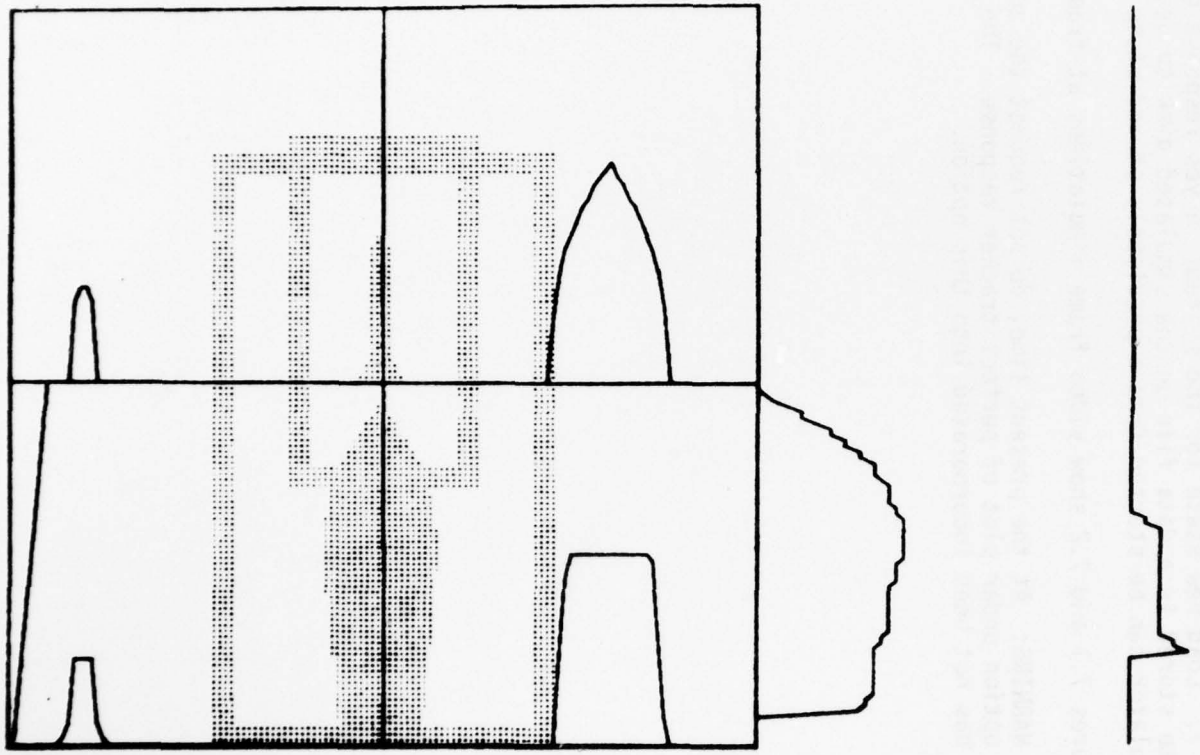


Figure 7.1. Video Tracker Simulation at Frame 19

TIME = 1.633 SEC.					98
RANGE = 21262.531 FT.					
	TARGET	PREDICTED	TRACKER		
AZIMUTH	0.0169	0.0169	0.0170	RAD.	
ELEVATION	0.3195	0.3193	0.3220	RAD.	
ROTATION ANGLE	0.0415	0.0415	0.0288	RAD.	
VIEW ANGLE	1.1539			RAD.	
ZOOM RATIO		1.80:1	1.80:1		

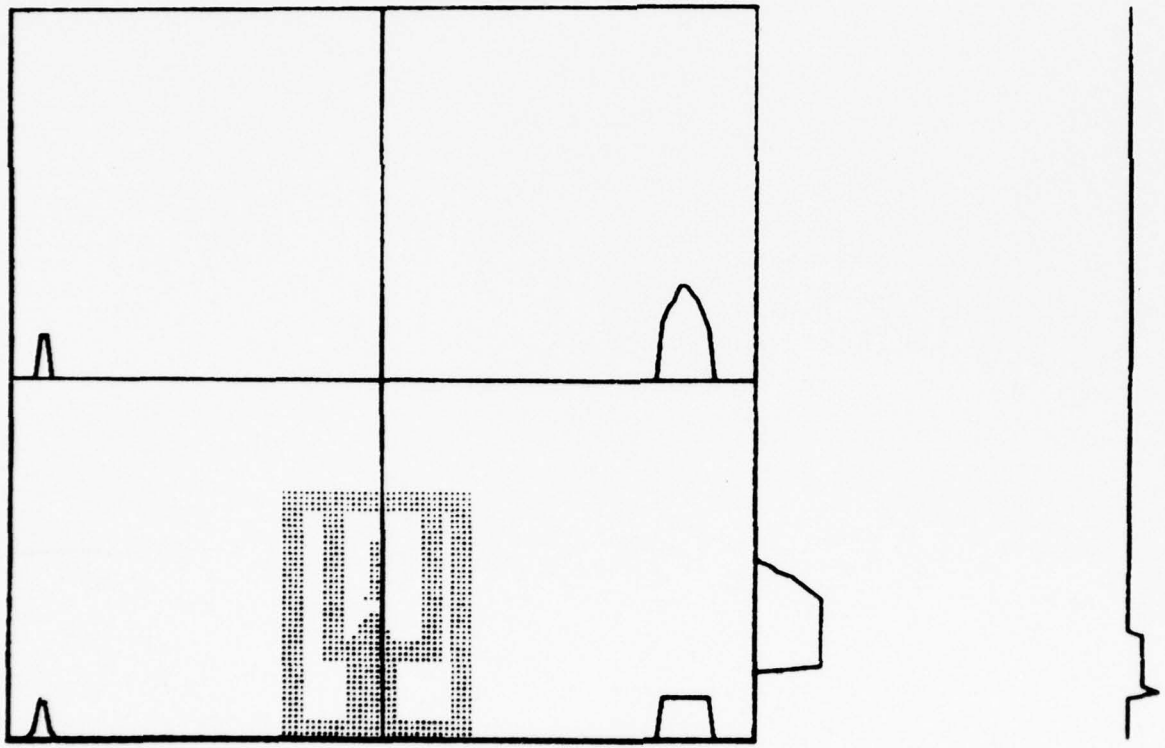


Figure 7.2. Video Tracker Simulation at Frame 98

SECTION 8

COMPUTER PROGRAMS

Following is the source code and some documentation for many of the useful programs developed at WSMR this summer.

PDP-11 MACRO SYSTEM PROGRAMS

```
        ;LPORIG          JULY 28,1977              MITCHELL
        ;DISPLAYS GRAY LEVEL PICTURES ON PRINTRONIX
        ;EXPECTS 240 POINTS PER ROW
        ;EACH POINT DISPLAYED USING 10 GRAY LEVELS CREATED USING
        ;A 3X3 MATRIX
        ;CAN DISPLAY OUTPUT OF DIGITIZER PROGRAMS
        .MCALL  ..V2..,.REGDEF,.CSIGEN,.READW,.CLOSE,.PRINT
        .MCALL  .EXIT,.WAIT,.TTYIN
        ..V2..
        .REGDEF
LPDAT=167772
LPSTAT=167770
FINISH: .PRINT  #RMSG
        .CLOSE  #3
        .EXIT
START:  .PRINT  #ANNON
        .PRINT  #ASK
        .CSIGEN #CORSPC,#FPRT
        CLR     BUFCNT
        .PRINT  #REQ1
        MOV     #BLACK,R1
        JSR     PC,ACCEPT
        .PRINT  #REQ2
        MOV     #INCR,R1
        JSR     PC,ACCEPT
LOOP1:  MOV     BUFCNT,R1
        ASL     R1
        ASL     R1
        ASL     R1
        ASL     R1
        SUB     BUFCNT,R1               ;15 BLOCKS PER BUFFER
        .READW  #AREA,#3,#BUF4,#7400,R1
        BCS     FINISH
        INC     BUFCNT
        MOV     #BUF4,R4
        MOV     #-40,LINES              ;NO OF LINES PER BUFFER
LOOP2:  MOV     #BUF1,R1                ;WRITES 1 LINE
        MOV     #BUF2,R2
        MOV     #BUF3,R3
        MOV     #-360,BYTES             ;ALLOWS FOR 240 POINTS PER LINE
LOOP3:  MOVB    (R4)+,R5                ;WRITES 1 CHARACTER
        BIC     #177400,R5
        SUB     BLACK,R5                ;BELOW #BLACK IS BLACK
        BGE     DIVIDE
        MOV     #TABLE-3,R5
        BR      LOOKUP
DIVIDE: MOV     #-1,R0                  ;DIVIDES R5 BY 10
AGAIN:  INC     R0
        SUB     INCR,R5
        BGE     AGAIN
        MOV     R0,R5
        CMP     #11,R5
        BLT     TOOBIG
        ASL     R5
        ADD     R0,R5           ;MULTIPLIES R5 BY 3
        ADD     #TABLE,R5
```

```
        BR      LOOKUP
        ADD     #TABLE,R5
LOOKUP: BIT     #1,BYTES
        BNE     SECOND
FIRST:  MOVB    (R5)+,(R1)
        MOVB    (R5)+,(R2)
        MOVB    (R5),(R3)
        BR      NEXT
SECOND: MOVB    (R5)+,R0
        ASL     R0
        ASL     R0
        ASL     R0
        BIS     #100,R0
        BISB    R0,(R1)+
        MOVB    (R5)+,R0
        ASL     R0
        ASL     R0
        ASL     R0
        BIS     #100,R0
        BISB    R0,(R2)+
        MOVB    (R5)+,R0
        ASL     R0
        ASL     R0
        ASL     R0
        BIS     #100,R0
        BISB    R0,(R3)+
NEXT:   INC     BYTES
        BEQ     LINEND
        JMP     LOOP3           ;BACK FOR ANOTHER CHARACTER
LINEND: MOVB    #005,(R1)
        MOVB    (R1),(R2)+
        MOVB    (R1)+,(R3)+
        MOVB    #012,(R1)
        MOVB    (R1),(R2)+
        MOVB    (R1)+,(R3)+
        SUB     #BUF1,R1
        SUB     #BUF2,R2
        SUB     #BUF3,R3
        MOV     #BUF1,R5
        JSR     PC,WRITIT
        MOV     R2,R1
        MOV     #BUF2,R5
        JSR     PC,WRITIT
        MOV     R3,R1
        MOV     #BUF3,R5
        JSR     PC,WRITIT
        INC     LINES
        BEQ     BUFEND
        JMP     LOOP2           ;BACK FOR ANOTHER LINE
BUFEND: JMP     LOOP1           ;BACK FOR ANOTHER BUFFER
WRITIT: BIT     #200,@#LPSTAT
        BEQ     WRITIT
```

```
        MOVB    (R5)+,@#LPDAT
        SOB     R1,WRITIT
        RTS     PC
ACCEPT: MOV     #BUF5,R2
INLOOP: .TTYIN  (R2)+
        CMPB    #12,R0
        BNE     INLOOP
        MOV     #BUF5,R2
        CLR     R5
        MOV     #3,R3
NXLOOP: MOVB    (R2)+,R4                ;ASSEMBLE INTO OCTAL NUMBER
        BIC     #177770,R4
        ASL     R5
        ASL     R5
        ASL     R5
        BIS     R4,R5
        SOB     R3,NXLOOP
        MOV     R5,(R1)
        RTS     PC
        .BYTE   7
        .BYTE   7
        .BYTE   7
TABLE:  .BYTE   7
        .BYTE   7
        .BYTE   6
        .BYTE   5
        .BYTE   7
        .BYTE   6
        .BYTE   5
        .BYTE   3
        .BYTE   6
        .BYTE   5
        .BYTE   3
        .BYTE   4
        .BYTE   4
        .BYTE   3
        .BYTE   4
        .BYTE   4
        .BYTE   3
        .BYTE   0
        .BYTE   4
        .BYTE   2
        .BYTE   0
        .BYTE   0
        .BYTE   2
        .BYTE   0
        .BYTE   0
        .BYTE   0
        .BYTE   0
        .BYTE   0
        .BYTE   0
        .BYTE   0
        .EVEN
FPRT:   .RAD50  /DAT/
        .WORD   0
        .WORD   0
        .WORD   0
```

```
AREA:    .BLKW   10
ANNON:   .ASCIZ  /LINE PRINTER 10 GRAY LEVELS,240 POINTS PER ROW/

ASK:     .ASCIZ  /ENTER INPUT FILE/
REQ1:    .ASCII  /ENTER OCTAL 3 DIGIT BLACK LEVEL /
         .BYTE 200
REQ2:    .ASCII  /ENTER OCTAL 3 DIGIT INCREMENT /
         BYTE 200
         .EVEN
BUFCNT:  .BLKW   1
KOUNT:   .BLKW   1
LINES:   .BLKW   1
BYTES:   .BLKW   1
BLACK:   .BLKW   1
INCR:    BLKW    1
CORSPC:  .BLKW   400              ;DEVICE HANDLER
BUF1:    .BLKW   240
BUF2:    .BLKW   240
BUF3:    .BLKW   240
BUF4:    .BLKW   7400
BUF5:    BLKW    50
         .END    START
```

```
          ;LPSMAL          AUGUST 8,1977          MITCHELL
          ;DISPLAYS GRAY LEVEL PICTURES ON PRINTRONIX
          ;EXPECTS 256 POINTS PER ROW
          ;EACH PONT DISPLAYED USING 22 GRAY LEVELS CREATED USING
          ;A 3X7 MATRIX (THIS ASSUMES RESOLUTION IS DOUBLE IN
          ;HORIZONTAL DIRECTION
          ;CAN DISPLAY OUTPUT OF TRANSPOSE OF DIGITIZER OUTPUT
           .MCALL   .V2.,.REGDEF,.CSIGEN,.READW,.CLOSE,.PRINT
           MCALL   .EXIT,.WAIT,.TTYIN
           .V2.
           .REGDEF
LPDAT=167772
LPSTAT=167770
FINISH:    .PRINT  #RMSG
           .CLOSE  #3
           .EXIT
START:     .PRINT  #ANNON
           .PRINT  #ASK
           .CSIGEN #CORSPC,#FPRT
           CLR     BUFCNT
           .PRINT  #REQ1
           MOV     #BLACK,R1
           JSR     PC,ACCEPT
           .PRINT  #REQ2
           MOV     #INCR,R1
           JSR     PC,ACCEPT
           .PRINT  #INMS
           .TTYIN
           CLR     INVERT
           CMPB    #111,R0
           BNE     LOOP1
           INC     INVERT
LOOP1:     MOV     BUFCNT,R1          ;WRITES ONE BUFFER
           ASL     R1
           ASL     R1                 ;4 BLOCKS PER BUFFER
           .READW  #AREA,#3,#BUF4,#2000,R1
           BCS     FINISH
           INC     BUFCNT
           MOV     #BUF4,R4
           MOV     #-10,LINES                 ;NO OF LINES PER BUFFER
LOOP2:     MOV     #BUF1,R1                   ;WRITES 1 LINE
           MOV     #-400,BYTES                ;ALLOWS FOR 256 POINTS PER LINE
LOOP3:     MOVB    (R4)+,R5                   ;WRITES 1 CHARACTER
           BIC     #177400,R5
           SUB     BLACK,R5                   ;BELOW #BLACK IS BLACK
           BGE     DIVIDE
           MOV     #TABLE-7,R5
           BR      LOOKUP
DIVIDE:    MOV     #-1,R0                     ;DIVIDES R5 BY 22
           TST     INCR
           BEQ     TOOBIG
AGAIN:     INC     R0
           SUB     INCR,R5
           BGE     AGAIN
           MOV     R0,R5
           CMP     #24,R5
```

```
        BLT     TOOBIG
        ASL     R5
        ASL     R5
        ASL     R5
        SUB     R0,R5           ;MULTIPLIES R3 BY 7
        ADD     #TABLE,R5
        BR      LOOKUP
TOOBIG: MOV     #214,R5
        ADD     #TABLE,R5
LOOKUP: TST     INVERT
        BEQ     LOOK2
        SUB     #TABLE-7,R5
        MOV     #TABLE+214,R0
        SUB     R5,R0
        MOV     R0,R5
LOOK2:  BIT     #1,BYTES
        BNE     SECOND
FIRST:  MOVB    (R5)+,(R1)
        MOV     R1,R2
        MOV     #6,R3
SIXMOR: ADD     #BUF2-BUF1,R2
        MOVB    (R5)+,(R2)
        SOB     R3,SIXMOR
        BR      NEXT
SECOND: MOV     R1,R2
        INC     R1
        MOV     #7,R3
SEVMOR: MOVB    (R5)+,R0
        ASL     R0
        ASL     R0
        ASL     R0
        BIS     #100,R0
        BISB    R0,(R2)
        ADD     #BUF2-BUF1,R2
        SOB     R3,SEVMOR
NEXT:   INC     BYTES
        BEQ     LINEND
        JMP     LOOP3           ;BACK FOR ANOTHER CHARACTER
LINEND: MOV     #7,R3
        MOV     R1,R2
MORE:   MOVB    #5,(R2)+        ;POINT PLOT MODE
        MOVB    #12,(R2)        ;END OF LINE
        ADD     #BUF2-BUF1-1,R2
        SOB     R3,MORE
        SUB     #BUF1-2,R1
        MOV     R1,R2
        MOV     #7,R3
        MOV     #BUF1,R0
ANOTH:  MOV     R2,R1
        MOV     R0,R5
        JSP     PC,WRITIT
        ADD     #BUF2-BUF1,R0
        SOB     R3,ANOTH
        INC     LINES
        BEQ     BUFEND
        JMP     LOOP2           ;BACK FOR ANOTHER LINE
BUFEND: JMP     LOOP1           ;BACK FOR ANOTHER BUFFER
```

```
WRITIT: BIT     #200,@#LPSTAT
        BEQ     WRITIT
        MOVB    (R5)+,@#LPDAT
        SOB     R1,WRITIT
        RTS     PC
ACCEPT: MOV     #BUF5,R2
INLOOP: .TTYIN  (R2)+
        CMPB    #12,R0
        BNE     INLOOP
        MOV     #BUF5,R2
        CLR     R5
        MOV     #3,R3
NXLOOP: MOVB    (R2)+,R4                ;ASSEMBLE INTO OCTAL NUMBER
        BIC     #177770,R4
        ASL     R5
        ASL     R5
        ASL     R5
        BIS     R4,R5
        SOB     R3,NXLOOP
        MOV     R5,(R1)
        RTS     PC
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   7
TABLE:  .BYTE   7
        .BYTE   5
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   5
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   3
        .BYTE   7
        .BYTE   7
        .BYTE   5
        .BYTE   7
        .BYTE   6
        .BYTE   7
        .BYTE   3
        .BYTE   7
        .BYTE   7
        .BYTE   5
        .BYTE   7
        .BYTE   6
        .BYTE   7
        .BYTE   3
        .BYTE   6
        .BYTE   7
```

```
.BYTE   5
.BYTE   3
.BYTE   6
.BYTE   7
.BYTE   3
.BYTE   6
.BYTE   7
.BYTE   5
.BYTE   3
.BYTE   6
.BYTE   5
.BYTE   3
.BYTE   6
.BYTE   3
.BYTE   5
.BYTE   3
.BYTE   6
.BYTE   5
.BYTE   3
.BYTE   6
.BYTE   3
.BYTE   4
.BYTE   3
.BYTE   6
.BYTE   5
.BYTE   3
.BYTE   6
.BYTE   3
.BYTE   4
.BYTE   3
.BYTE   6
.BYTE   5
.BYTE   3
.BYTE   4
.BYTE   3
.BYTE   4
.BYTE   5
.BYTE   3
.BYTE   4
.BYTE   3
.BYTE   4
.BYTE   3
.BYTE   4
.BYTE   5
.BYTE   2
.BYTE   4
.BYTE   3
.BYTE   4
.BYTE   3
.BYTE   4
.BYTE   1
.BYTE   2
.BYTE   4
.BYTE   2
```

```
.BYTE   4
.BYTE   3
 BYTE   4
.BYTE   1
.BYTE   2
.BYTE   4
 BYTE   2
.BYTE   4
.BYTE   1
.BYTE   4
.BYTE   1
.BYTE   2
.BYTE   4
.BYTE   2
.BYTE   4
.BYTE   1
.BYTE   4
.BYTE   1
.BYTE   2
.BYTE   0
.BYTE   2
.BYTE   0
.BYTE   1
.BYTE   4
.BYTE   1
.BYTE   2
.BYTE   0
.BYTE   2
.BYTE   0
.BYTE   1
.BYTE   4
.BYTE   0
.BYTE   2
.BYTE   0
.BYTE   0
.BYTE   0
.BYTE   1
.BYTE   4
.BYTE   0
.BYTE   2
.BYTE   0
.BYTE   0
.BYTE   0
.BYTE   1
.BYTE   4
.BYTE   0
.BYTE   0
.BYTE   0
.BYTE   0
.BYTE   0
.BYTE   1
.BYTE   0
.BYTE   0
.BYTE   0
.BYTE   0
.BYTE   0
```

```
        .BYTE   0
        .BYTE   0
        BYTE    0
        .BYTE   0
        .BYTE   0
        .BYTE   0

        .EVEN
FPRT:   .RAD50  /DAT/
        WORD    0
        WORD    0
        .WORD   0
AREA:   .BLKW   10
RMSG:   .ASCIZ  /THAT'S ALL/
ANNON:  .ASCIZ  /LINE PRINTER 32 GRAY LEVELS 256 POINTS PER ROW/

ASK:    .ASCIZ  /ENTER INPUT FILE/
REQ1:   .ASCII  /ENTER OCTAL 3 DIGIT BLACK LEVEL /
        BYTE 200
REQ2:   .ASCII  /ENTER OCTAL 3 DIGIT INCREMENT /
        .BYTE 200
INMS:   .ASCIZ  /TYPE AN "I" TO INVERT VIDEO/
        .EVEN
BUFCNT: .BLKW   1
KOUNT:  .BLKW   1
INVERT: .BLKW   1
LINES:  .BLKW   1
BYTES:  .BLKW   1
BLACK:  .BLKW   1
INCR:   .BLKW   1
CORSPC: .BLKW   400             ;DEVICE HANDLER
BUF1:   .BLKW   240
BUF2:   .BLKW   240
BUF3:   .BLKW   1240
BUF4:   .BLKW   2000
BUF5:   .BLKW   50
        .END    START
```

```
        ;LPLARG          JULY 28,1977              MITCHELL
        ;DISPLAYS GRAY LEVEL PICTURES ON PRINTRONIX
        ;EXPECTS 512 POINTS PER ROW
        ;REQIRES TWO PAGES -- LEFT AND RIGHT
        ;EACH POINT DISPLAYED USING 22 GRAY LEVELS CREATED USING
        ;A 3X7 MATRIX (THIS ASSUMES RESOLUTION IS DOUBLE IN
        ;HORIZONTAL DIRECTION.
        ;CAN DISPLAY OUTPUT OF TRANSPOSE OF DIGITIZER OUTPUT.
        .MCALL  ..V2..,.REGDEF,.CSIGEN,.READW,.CLOSE,.PRINT
        .MCALL  .EXIT,.WAIT,.TTYIN
        ..V2..
        .REGDEF
LPDAT=167772
LPSTAT=167770
FINISH: .PRINT  #RMSG
        .CLOSE  #3
        .EXIT
START:  .PRINT  #ANNON
        .PRINT  #ASK
        .CSIGEN #CORSPC,#FPRT
        CLR     BUFCNT
        .PRINT  #REQ1
        MOV     #BLACK,R1
        JSR     PC,ACCEPT
        .PRINT  #REQ2
        MOV     #INCR,R1
        JSR     PC,ACCEPT
        .PRINT  #INMS
        .TTYIN
        CLR     INVERT
        CMPB    #111,R0
        BNE     PASS
        INC     INVERT
PASS:   CLR     RIGHT
        BR      LOOP1
HOLD:   BIT     #200,@#LPSTAT
        BEQ     HOLD
        MOVB    #14,@#LPDAT     ;FORM FEED
        TST     RIGHT
        BNE     FINISH
        INC     RIGHT           ;PRINT RIGHT HALF OF PICTURE
        CLR     BUFCNT
LOOP1:  MOV     BUFCNT,R1       ;WRITES ONE BUFFER
        ASL     R1
        ASL     R1
        ASL     R1
        ASL     R1              ;16 BLOCKS PER BUFFER
        .READW  #AREA,#3,#BUF4,#10000,R1
        BCS     HOLD
        INC     BUFCNT
        MOV     #BUF4,R4
        TST     RIGHT
        BEQ     SKIP
        ADD     #400,R4         ;RIGHT HALF OF EACH BUFFER
SKIP:   MOV     #-20,LINES      ;NO OF LINES PER BUFFER
LOOP2:  MOV     #BUF1,R1                ;WRITES 1 LINE
```

```
        MOV     #-400,BYTES             ;ALLOWS FOR 256 POINTS PER LINE
LOOP3:  MOVB    (R4)+,R5                ;WRITES 1 CHARACTER
        BIC     #177400,R5
        SUB     BLACK,R5                ;BELOW #BLACK IS BLACK
        BGE     DIVIDE
        MOV     #TABLE-7,R5
        BR      LOOKUP
DIVIDE: MOV     #-1,R0                  ;DIVIDES R5 BY 22
AGAIN:  INC     R0
        SUB     INCR,R5
        BGE     AGAIN
        MOV     R0,R5
        CMP     #24,R5
        BLT     TOOBIG
        ASL     R5
        ASL     R5
        ASL     R5
        SUB     R0,R5               ;MULTIPLIES R5 BY 7
        ADD     #TABLE,R5
        BR      LOOKUP
TOOBIG: MOV     #214,R5
        ADD     #TABLE,R5
LOOKUP: TST     INVERT
        BEQ     LOOK2
        SUB     #TABLE-7,R5
        MOV     #TABLE+214,R0
        SUB     R5,R0
        MOV     R0,R5
LOOK2:  BIT     #1,BYTES
        BNE     SECOND
FIRST:  MOVB    (R5)+,(R1)
        MOV     R1,R2
        MOV     #6,R3
SIXMOR: ADD     #BUF2-BUF1,R2
        MOVB    (R5)+,(R2)
        SOB     R3,SIXMOR
        BR      NEXT
SECOND: MOV     R1,R2
        INC     R1
        MOV     #7,R3
SEVMOR: MOVB    (R5)+,R0
        ASL     R0
        ASL     R0
        ASL     R0
        BIS     #100,R0
        BISB    R0,(R2)
        ADD     #BUF2-BUF1,R2
        SOB     R3,SEVMOR
NEXT:   INC     BYTES
        BEQ     LINEND
        JMP     LOOP3               ;BACK FOR ANOTHER CHARACTER
LINEND: MOV     #7,R3
        MOV     R1,R2
MORE:   MOVB    #5,(R2)+            ;POINT PLOT MODE
        MOVB    #12,(R2)            ;END OF LINE
        ADD     #BUF2-BUF1-1,R2
        SOB     R3,MORE
```

```
        MOV     R1,R2-2,R1
        MOV     #7,R3
        MOV     #BUF1,R0
ANOTH:  MOV     R2,R1
        MOV     R0,R5
        JSR     PC,WRITIT
        ADD     #BUF2-BUF1,R0
        SOB     R3,ANOTH
        INC     LINES
        BEQ     BUFEND
        ADD     #400,R4             ;SKIP NEXT LINE HALF
        JMP     LOOP2               ;BACK FOR ANOTHER LINE
BUFEND: JMP     LOOP1               ;BACK FOR ANOTHER BUFFER
WRITIT: BIT     #200,@#LPSTAT
        BEQ     WRITIT
        MOVB    (R5)+,@#LPDAT
        SOB     R1,WRITIT
        RTS     PC
ACCEPT: MOV     #BUF5,R2
INLOOP: .TTYIN  (R2)+
        CMPB    #12,R0
        BNE     INLOOP
        MOV     #BUF5,R2
        CLR     R5
        MOV     #3,R3
NXLOOP: MOVB    (R2)+,R4                ;ASSEMBLE INTO OCTAL NUMBER
        BIC     #177770,R4
        ASL     R5
        ASL     R5
        ASL     R5
        BIS     R4,R5
        SOB     R3,NXLOOP
        MOV     R5,(R1)
        RTS     PC
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   7
TABLE:  .BYTE   7
        .BYTE   5
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   5
        .BYTE   7
        .BYTE   7
        .BYTE   7
        .BYTE   3
        .BYTE   7
```

```
BYTE    7
BYTE    5
BYTE    7
BYTE    6
BYTE    7
BYTE    3
BYTE    7
BYTE    7
BYTE    5
BYTE    7
BYTE    6
BYTE    7
BYTE    3
BYTE    6
BYTE    7
BYTE    5
BYTE    3
BYTE    6
BYTE    7
BYTE    3
BYTE    6
BYTE    7
BYTE    5
BYTE    3
BYTE    6
BYTE    5
BYTE    3
BYTE    6
BYTE    3
BYTE    5
BYTE    3
BYTE    6
BYTE    5
BYTE    3
BYTE    6
BYTE    3
BYTE    4
BYTE    3
BYTE    6
BYTE    5
BYTE    3
BYTE    6
BYTE    3
BYTE    4
BYTE    3
BYTE    6
BYTE    5
BYTE    3
BYTE    4
BYTE    3
BYTE    4
BYTE    3
BYTE    4
BYTE    5
BYTE    3
```

```
BYTE    4
BYTE    3
BYTE    4
BYTE    3
BYTE    4
BYTE    5
BYTE    2
BYTE    4
BYTE    3
BYTE    4
BYTE    3
BYTE    4
BYTE    1
BYTE    2
BYTE    4
BYTE    2
BYTE    4
BYTE    3
BYTE    4
BYTE    1
BYTE    2
BYTE    4
BYTE    2
BYTE    4
BYTE    1
BYTE    4
BYTE    1
BYTE    2
BYTE    4
BYTE    2
BYTE    4
BYTE    1
BYTE    4
BYTE    1
BYTE    2
BYTE    0
BYTE    2
BYTE    0
BYTE    1
BYTE    4
BYTE    1
BYTE    2
BYTE    0
BYTE    2
BYTE    0
BYTE    1
BYTE    4
BYTE    0
BYTE    2
BYTE    0
BYTE    0
BYTE    0
BYTE    1
BYTE    4
BYTE    0
BYTE    2
BYTE    0
```

```
        .BYTE   0
        .BYTE   0
        .BYTE   1
        .BYTE   4
        .BYTE   0
        .BYTE   0
        .BYTE   0
        .BYTE   0
        .BYTE   0
        .BYTE   1
        .BYTE   0
        .BYTE   0
        .BYTE   0
        .BYTE   0
        .BYTE   0
        .BYTE   0
        .BYTE   0
        .BYTE   0
        .BYTE   0
        .BYTE   0

        .EVEN
FPRT:   .RAD50  /DAT/
        .WORD   0
        .WORD   0
        .WORD   0
AREA:   .BLKW   10
RMSG:   .ASCIZ  /THAT'S ALL/
ANNON:  .ASCIZ  /LINE PRINTER 22 GRAY LEVELS, 512 POINTS PER ROW/

ASK:    .ASCIZ  /ENTER INPUT FILE/
REQ1:   .ASCII  /ENTER OCTAL 3 DIGIT BLACK LEVEL /
        .BYTE 200
REQ2:   .ASCII  /ENTER OCTAL 3 DIGIT INCREMENT /
        .BYTE 200
INMS:   .ASCIZ  /TYPE AN "I" TO INVERT VIDEO/
        .EVEN
BUFCNT: .BLKW   1
KOUNT:  .BLKW   1
RIGHT:  .BLKW   1
INVERT: .BLKW   1
LINES:  .BLKW   1
BYTES:  .BLKW   1
BLACK:  .BLKW   1
INCR:   .BLKW   1
CORSPC: .BLKW   400             ;DEVICE HANDLER
BUF1:   .BLKW   240
BUF2:   .BLKW   240
BUF3:   .BLKW   1240
BUF4:   .BLKW   10000
BUF5:   .BLKW   50
        .END    START
```

```
          ;PROGRAM DIG5F1              AUG 9, 1977
          ;STORES DIGITIZED DATA FROM VIDEO FIELD ONE IN A FLOPPY
          ;DISK FILE.  DATA IS STORED BY COLUMNS:  512 COLUMNS,
          ;240 PIXELS PER COLUMN.
           .MCALL   .V2 ..REGDEF, .ENTER, .WRITE, .CLOSE, .PRINT
          .MCALL   .EXIT, .FETCH, .WAIT
           .V2
           REGDEF
ADDR=167772
DATA=167774
STAT=167770
START:          .FETCH   #CORSPC,#FPRT
          BCS     BADFET      ;FETCH ERROR
           .ENTER  #AREA,#0,#FPRT,#-1
          BCS     BADEN       ;AN ENTER ERROR
          MOV     #1,FBUF     ;SET BUFFER FLAG
          CLR     KOUNT       ;CLEAR BUFFER COUNTER
        CLR     @#STAT
        MOV     #1,@#ADDR
        MOV     #2001,@#ADDR
        CLR     @#ADDR
        MOV     #-1,R3
          MOV     #BUF1,R2    ;PREPARE TO FILL BUF1
        JSR     PC,FILL     ;FILL BUF1
          MOV     #10,LIMIT   ;NO. OF BUFFERS TO READ
        BR      BEGIN
BADFET:     .PRINT #FMSG
           .EXIT
BADEN:      .PRINT #EMSG
           .EXIT
BADWRT:      .PRINT #WMSG
        .EXIT

FINISH:     .CLOSE #0
           .EXIT

BEGIN:      CMP     LIMIT,KOUNT ;IS FLOPPY FULL
           BEQ     FINISHED    ;ALL DATA HAS BEEN READ
FIRST:      CMP     FBUF,#0     ;IS BUF 2 TO BE WRITTEN
           BNE     OTHER       ;IF NOT GO TO WRITE #2
        MOV     KOUNT,R1;READY TO MULTIPLY
        MUL     #36,R1   ;BY 30
                  .WRITE   #AREA,#0,#BUF2,#17000,R1
        BCS     BADWRT
          INC     KOUNT       ;COUNT BUFFER WRITTEN
          MOV     #BUF1,R2    ;PREPARE TO CALL FILL
         JSR     PC,FILL     ;FILL BUF1
          MOV     #1,FBUF     ;SET FLAG TO INDICATE BUF1 FILLED
           .WAIT  #0          ;WAIT TILL FINISHED WRITING
          JMP     BEGIN       ; GO AGAIN
OTHER:  MOV KOUNT,R1;PREPARE TO MULTPY
          MUL   #36,R1;MULTPLY
    .WRITE  #AREA,#0,#BUF1,#17000,R1
        BCS     BADWRT
          INC     KOUNT       ;COUNT BUFFER WRITTEN
```

```
            MOV     #BUF2,R2        ;PREPARE TO CALL FILL
            CLR     FBUF            ;INDICATE BUF2 FILLED
            JSR     PC,FILL         ;FILL BUF2
            .WAIT #0                ;WAIT TILL BUF1 IS WRITTEN
            JMP     BEGIN           ;START AGAIN

FILL:     CLR     KOLUM
          CLR     R5                ;INITIALIZE ROWS
COLUMNS.  INC     R3                ;PROCESS COLUMNS
          INC     KOLUM             ;COUNT COLUMNS
          MOV     R3,@#ADDR         ;LOAD HORIZONTAL
          MOV     #1,@#STAT         ;STROBE HORIZONTAL
          CLR     @#STAT   ;PREPARE FOR NEXT COORD
ROWS:     INC     R5                ;PROCESS ROWS
          MOV     R5,@#ADDR         ;ROW COORDINATE
          MOV     #2,@#STAT
          CLR @#STAT                ;STROBE VERTICAL
          BIS     #1000,@#ADDR      ;SET LEND
          MOV     #100000,@#ADDR    ;EXECUTE
          CLR     @#ADDR            ;CLEAR EXECUTE
WAIT:     BIT     #40000,@#DATA
          BNE     TEST2
          BIT     #1,R5
          BNE     WAIT
          BR      GO
TEST2:    BIT     #1,R5
          BEQ     WAIT
GO:       MOVB    @#DATA,(R2)+
          CMP     #354,R5           ;HAS LAST ROW BEEN PROCESSED
          BNE     ROWS              ;IF NOT GET OTHER ROWS
          CLR     (R2)+
          CLR     (R2)+             ;ADD 4 BYTES OF ZEROS
          CLR     R5
          CMP     #100,KOLUM
          BNE     COLUMNS
          RTS     PC                ;RETURN FROM SUBROUTINE

KOLUM:FPRT:       .RAD50  /DX1/
          .RAD50  /ORIG   /
          .RAD50 /DAT/
AREA:       BLKW  10
FMSG:       .ASCIZ /BAD FETCH/
EMSG:       .ASCIZ /BAD ENTER/
WMSG:       .ASCIZ /WRITE ERROR/
          .EVEN

PARM:     .WORD 0

CORSPC:     .BLKW     400   ;DEVICE HANDLER
          .WORD  0
FBUF:       .BLKW  1
LIMIT:      .BLKW  1
KOUNT:  .BLKW  1
BUF1:
          .BLKW 17000
END1:
BUF2:
```

```
        .BLKW 17000
END2:
        .END START
```

```
          ;PROGRAM DIG5F2            AUG 9, 1977
          ;STORES DIGITIZED DATA FROM VIDEO FIELD TWO IN A FLOPPY
          ;DISK FILE.  DATA IS STORED BY COLUMNS: 512 COLUMNS,
          ;240 PIXELS PER COLUMN.
          .MCALL  .V2.,.REGDEF,.ENTER, WRITE,.CLOSE,.PRINT
          .MCALL  .EXIT,.FETCH,.WAIT
          .V2.
          .REGDEF
ADDR=167772
DATA=167774
STAT=167770
START:          .FETCH   #CORSPC,#FPRT
          BCS     BADFET     ;FETCH ERROR
          .ENTER  #AREA,#0,#FPRT,#-1
          BCS     BADEN        ;AN ENTER ERROR
          MOV     #1,FBUF       ;SET BUFFER FLAG
          CLR     KOUNT         ;CLEAR BUFFER COUNTER
        CLR     @#STAT
        MOV     #1,@#ADDR
        MOV     #2001,@#ADDR
        CLR     @#ADDR
        MOV     #-1,R3
          MOV     #BUF1,R2      ;PREPARE TO FILL BUF1
        JSR     PC,FILL       ;FILL BUF1
          MOV     #10,LIMIT     ;NO. OF BUFFERS TO READ
        BR      BEGIN
BADFET:    .PRINT #FMSG
          .EXIT
BADEN:     .PRINT #EMSG
          .EXIT
BADWRT:     .PRINT #WMSG
         .EXIT

FINISH:     CLOSE #0
          .EXIT

BEGIN:     CMP     LIMIT,KOUNT   ;IS FLOPPY FULL
          BEQ     FINISHED      ;ALL DATA HAS BEEN READ
FIRST:     CMP     FBUF,#0       ;IS BUF 2 TO BE WRITTEN
          BNE     OTHER         ;IF NOT GO TO WRITE #2
        MOV     KOUNT,R1;READY TO MULTIPLY
        MUL     #36,R1   ;BY 30
                  .WRITE   #AREA,#0,#BUF2,#17000,R1
        BCS     BADWRT
          INC     KOUNT         ;COUNT BUFFER WRITTEN
          MOV     #BUF1,R2      ;PREPARE TO CALL FILL
         JSR     PC,FILL       ;FILL BUF1
          MOV     #1,FBUF       ;SET FLAG TO INDICATE BUF1 FILLED
          .WAIT   #0            ;WAIT TILL FINISHED WRITING
          JMP     BEGIN         ; GO AGAIN
OTHER:   MOV KOUNT,R1;PREPARE TO MULTPY
          MUL   #36,R1;MULTPLY
    .WRITE  #AREA,#0,#BUF1,#17000,R1
        BCS     BADWRT
          INC     KOUNT         ;COUNT BUFFER WRITTEN
```

```
ALL FILL
          CLR     FBUF            ;INDICATE BUF2 FILLED
          JSR     PC,FILL         ;FILL BUF2
          .WAIT #0                ;WAIT TILL BUF1 IS WRITTEN
          JMP     BEGIN           ;START AGAIN

FILL:     CLR     KOLUM
          MOV     #400,R5                 ;INITIALIZE ROWS
COLUMNS:  INC     R3                      ;PROCESS COLUMNS
          INC     KOLUM                   ;COUNT COLUMNS
          MOV     R3,@#ADDR               ;LOAD HORIZONTAL
          MOV     #1,@#STAT               ;STROBE HORIZONTAL
          CLR     @#STAT  ;PREPARE FOR NEXT COORD
ROWS:     INC     R5                      ;PROCESS ROWS
          MOV     R5,@#ADDR               ;ROW COORDINATE
          MOV     #2,@#STAT
          CLR @#STAT                      ;STROBE VERTICAL
          BIS     #1000,@#ADDR            ;SET LEND
          MOV     #100000,@#ADDR          ;EXECUTE
          CLR     @#ADDR                  ;CLEAR EXECUTE
WAIT:     BIT     #40000,@#DATA
          BNE     TEST2
          BIT     #1,R5
          BNE     WAIT
          BR      GO
TEST2:    BIT     #1,R5
          BEQ     WAIT
GO:       MOVB    @#DATA,(R2)+
          CMP     #754,R5                 ;HAS LAST ROW BEEN PROCESSED
          BNE     ROWS                    ;IF NOT GET OTHER ROWS
          CLR     (R2)+
          CLR     (R2)+                   ;ADD 4 BYTES OF ZEROS
          MOV     #400,R5
          CMP     #100,KOLUM
          BNE     COLUMNS
          RTS     PC                      ;RETURN FROM SUBROUTINE

KOLUM:FPRT:       .RAD50  /DX1/
          .RAD50  /ORIG  /
          .RAD50 /DAT/
AREA:       .BLKW  10
FMSG:       .ASCIZ /BAD FETCH/
EMSG:       .ASCIZ /BAD ENTER/
WMSG:       .ASCIZ /WRITE ERROR/
          .EVEN

PARM:     .WORD 0

CORSPC:     .BLKW     400   ;DEVICE HANDLER
          .WORD  0
FBUF:       .BLKW  1
LIMIT:      .BLKW  1
KOUNT:  .BLKW  1
BUF1:
          .BLKW 17000
END1:
```

```
BUF2:
        .BLKW 17000
END2:
        .END START
```

```
          ;PROGRAM DIG2F1              AUG 9, 1977
          ;STORES DIGITIZED DATA FROM VIDEO FIELD ONE IN A FLOPPY
          ;DISK FILE.  DATA IS STORED BY COLUMNS:  256 COLUMNS,
          ;240 PIXELS PER COLUMN
          .MCALL  .V2.  ,.REGDEF,.ENTER,.WRITE,.CLOSE,.PRINT
          .MCALL  .EXIT,.FETCH,.WAIT
           .V2.
           REGDEF
ADDR=167772
DATA=167774
STAT=167770
START:          .FETCH   #CORSPC,#FPRT
          BCS     BADFET      ;FETCH ERROR
          .ENTER  #AREA,#0,#FPRT,#-1
          BCS     BADEN       ;AN ENTER ERROR
          MOV     #1,FBUF     ;SET BUFFER FLAG
          CLR     KOUNT       ;CLEAR BUFFER COUNTER
        CLR     @#STAT
        MOV     #1,@#ADDR
        MOV     #2001,@#ADDR
        CLR     @#ADDR
        MOV     #-2,R3
          MOV     #BUF1,R2    ;PREPARE TO FILL BUF1
        JSR     PC,FILL     ;FILL BUF1
          MOV     #4,LIMIT    ;NO. OF BUFFERS TO READ
        BR      BEGIN
BADFET:     .PRINT #FMSG
          .EXIT
BADEN:      .PRINT #EMSG
          .EXIT
BADWRT:     .PRINT #WMSG
         .EXIT

FINISH:     .CLOSE #0
          .EXIT

BEGIN:      CMP     LIMIT,KOUNT ;IS FLOPPY FULL
          BEQ     FINISHED    ;ALL DATA HAS BEEN READ
FIRST:      CMP     FBUF,#0     ;IS BUF 2 TO BE WRITTEN
          BNE     OTHER       ;IF NOT GO TO WRITE #2
        MOV     KOUNT,R1;READY TO MULTIPLY
        MUL     #36,R1   ;BY 30
                  .WRITE    #AREA,#0,#BUF2,#17000,R1
        BCS     BADWRT
          INC     KOUNT       ;COUNT BUFFER WRITTEN
          MOV     #BUF1,R2    ;PREPARE TO CALL FILL
         JSR     PC,FILL     ;FILL BUF1
          MOV     #1,FBUF     ;SET FLAG TO INDICATE BUF1 FILLED
          .WAIT   #0          ;WAIT TILL FINISHED WRITING
          JMP     BEGIN       ; GO AGAIN
OTHER:  MOV KOUNT,R1;PREPARE TO MULTPY
          MUL   #36,R1;MULTPLY
    .WRITE  #AREA,#0,#BUF1,#17000,R1
        BCS     BADWRT
          INC     KOUNT       ;COUNT BUFFER WRITTEN
```

```
          MOV     #BUF2,R2        ;PREPARE TO CALL FILL
          CLR     FBUF            ;INDICATE BUF2 FILLED
          JSR     PC,FILL         ;FILL BUF2
          .WAIT #0                ;WAIT TILL BUF1 IS WRITTEN
          JMP     BEGIN           ;START AGAIN

FILL:     CLR     KOLUM
          CLR     R5                      ;INITIALIZE ROWS
COLUMNS:  ADD     #2,R3                   ;PROCESS COLUMNS
          INC     KOLUM                   ;COUNT COLUMNS
          MOV     R3,@#ADDR               ;LOAD HORIZONTAL
          MOV     #1,@#STAT               ;STROBE HORIZONTAL
          CLR     @#STAT  ;PREPARE FOR NEXT COORD
ROWS:     INC     R5                      ;PROCESS ROWS
          MOV     R5,@#ADDR               ;ROW COORDINATE
          MOV     #2,@#STAT
          CLR @#STAT                      ;STROBE VERTICAL
          BIS     #1000,@#ADDR            ;SET LEND
          MOV     #100000,@#ADDR          ;EXECUTE
          CLR     @#ADDR                  ;CLEAR EXECUTE
WAIT:     BIT     #40000,@#DATA
          BNE     TEST2
          BIT     #1,R5
          BNE     WAIT
          BR      GO
TEST2:    BIT     #1,R5
          BEQ     WAIT
GO:       MOVB    @#DATA,(R2)+
          CMP     #354,R5                 ;HAS LAST ROW BEEN PROCESSED
          BNE     ROWS                    ;IF NOT GET OTHER ROWS
          CLR     (R2)+
          CLR     (R2)+                   ;ADD 4 BYTES OF ZEROS
          CLR     R5
          CMP     #100,KOLUM
          BNE     COLUMNS
          RTS     PC                      ;RETURN FROM SUBROUTINE

FPRT:       .RAD50  /DX1/
          .RAD50  /ORIG  /
          .RAD50 /DAT/
AREA:       .BLKW  10
FMSG:       .ASCIZ /BAD FETCH/
EMSG:       .ASCIZ /BAD ENTER/
WMSG:       .ASCIZ /WRITE ERROR/
          .EVEN

PARM:     .WORD 0

CORSPC:     .BLKW      400  ;DEVICE HANDLER
KOLUM:    .WORD  0
FBUF:       .BLKW  1
LIMIT:      .BLKW  1
KOUNT:  .BLKW  1
BUF1:
          .BLKW 17000
END1:
```

```
BUF2:
        .BLKW 17000
END2:
        .END START
```

```
          ;PROGRAM DIG2F2              AUG 9, 1977
          ;STORES DIGITIZED DATA FROM VIDEO FIELD TWO IN A FLOPPY
          ;DISK FILE.  DATA IS STORED BY COLUMNS:  256 COLUMNS,
          ;240 PIXELS PER COLUMN.
          .MCALL ..V2..,.REGDEF,.ENTER,.WRITE,.CLOSE,.PRINT
          .MCALL  .EXIT,.FETCH,.WAIT
          ..V2..
          .REGDEF
ADDR=167772
DATA=167774
STAT=167770
START:          .FETCH   #CORSPC,#FPRT
          BCS     BADFET     ;FETCH ERROR
          .ENTER  #AREA,#0,#FPRT,#-1
          BCS     BADEN        ;AN ENTER ERROR
          MOV     #1,FBUF       ;SET BUFFER FLAG
          CLR     KOUNT         ;CLEAR BUFFER COUNTER
        CLR     @#STAT
        MOV     #1,@#ADDR
        MOV     #2001,@#ADDR
        CLR     @#ADDR
        MOV     #-2,R3
          MOV     #BUF1,R2      ;PREPARE TO FILL BUF1
        JSR     PC,FILL       ;FILL BUF1
          MOV     #4,LIMIT      ;NO. OF BUFFERS TO READ
        BR      BEGIN
BADFET:   .PRINT #FMSG
          .EXIT
BADEN:    .PRINT #EMSG
          .EXIT
BADWRT:    .PRINT #WMSG
        .EXIT

FINISH:   .CLOSE #0
          .EXIT

BEGIN:    CMP     LIMIT,KOUNT   ;IS FLOPPY FULL
          BEQ     FINISHED      ;ALL DATA HAS BEEN READ
FIRST:    CMP     FBUF,#0       ;IS BUF 2 TO BE WRITTEN
          BNE     OTHER         ;IF NOT GO TO WRITE #2
        MOV     KOUNT,R1;READY TO MULTIPLY
        MUL     #36,R1   ;BY 30
                  .WRITE   #AREA,#0,#BUF2,#17000,R1
        BCS     BADWRT
          INC     KOUNT         ;COUNT BUFFER WRITTEN
          MOV     #BUF1,R2      ;PREPARE TO CALL FILL
         JSR     PC,FILL       ;FILL BUF1
          MOV     #1,FBUF       ;SET FLAG TO INDICATE BUF1 FILLED
          .WAIT   #0            ;WAIT TILL FINISHED WRITING
          JMP     BEGIN         ; GO AGAIN
OTHER:  MOV KOUNT,R1;PREPARE TO MULTPY
          MUL   #36,R1;MULTPLY
    .WRITE  #AREA,#0,#BUF1,#17000,R1
        BCS     BADWRT
          INC     KOUNT         ;COUNT BUFFER WRITTEN
          MOV     #BUF2,R2      ;PREPARE TO CALL FILL
```

```
          CLR     FBUF            ;INDICATE BUF2 FILLED
          JSR     PC,FILL         ;FILL BUF2
          .WAIT #0                ;WAIT TILL BUF1 IS WRITTEN
          JMP     BEGIN           ;START AGAIN

FILL:     CLR     KOLUM
          MOV     #400,R5             ;INITIALIZE ROWS
COLUMNS:  ADD     #2,R3               ;PROCESS COLUMNS
          INC     KOLUM               ;COUNT COLUMNS
          MOV     R3,@#ADDR           ;LOAD HORIZONTAL
          MOV     #1,@#STAT           ;STROBE HORIZONTAL
          CLR     @#STAT   ;PREPARE FOR NEXT COORD
ROWS:     INC     R5                  ;PROCESS ROWS
          MOV     R5,@#ADDR           ;ROW COORDINATE
          MOV     #2,@#STAT
          CLR @#STAT                  ;STROBE VERTICAL
          BIS     #1000,@#ADDR        ;SET LEND
          MOV     #100000,@#ADDR      ;EXECUTE
          CLR     @#ADDR              ;CLEAR EXECUTE
WAIT:     BIT     #40000,@#DATA
          BNE     TEST2
          BIT     #1,R5
          BNE     WAIT
          BR      GO
TEST2:    BIT     #1,R5
          BEQ     WAIT
GO:       MOVB    @#DATA,(R2)+
          CMP     #754,R5             ;HAS LAST ROW BEEN PROCESSED
          BNE     ROWS                ;IF NOT GET OTHER ROWS
          CLR     (R2)+
          CLR     (R2)+               ;ADD 4 BYTES OF ZEROS
          MOV     #400,R5
          CMP     #100,KOLUM
          BNE     COLUMNS
          RTS     PC                  ;RETURN FROM SUBROUTINE

FPRT:       .RAD50  /DX1/
          .RAD50  /ORIG  /
          .RAD50 /DAT/
AREA:       .BLKW  10
FMSG:       .ASCIZ /BAD FETCH/
EMSG:       .ASCIZ /BAD ENTER/
WMSG:       .ASCIZ /WRITE ERROR/
          .EVEN

PARM:     .WORD 0

CORSPC:       .BLKW      400   ;DEVICE HANDLER
KOLUM:    .WORD  0
FBUF:       .BLKW  1
LIMIT:      .BLKW  1
KOUNT:    .BLKW  1
BUF1:
          .BLKW 17000
END1:
BUF2:
          .BLKW 17000
```

```
END2:
        .END START
```

```
        ;PROGRAM MAG256          AUG 10,1977          MACHUCA
        ;COPIES PICTURE FILES CONTAINING 256 PIXELS PER
        ;ROW FROM FLOPPY DISKS ONTO 7-TRACK MAGNETIC TAPE.
        ;THE 8-BIT PIXELS ARE STACKED ONTO THE CONSECUTIVE LINES
        ;OF THE TAPE SO THAT THE FIRST SIX BITS OF PIXEL ONE ARE
        ;RECORDED ON THE FIRST SIX TRACKS OF THE FIRST LINE, THE
        ;SEVENTH AND EIGHTH BITS OF PIXEL ONE ARE RECORDED ON THE
        ;FIRST TWO TRACKS OF THE SECOND LINE, ETC., UNTIL ALL
        ;PIXELS ARE PACKED ONTO THE TAPE.  THE SEVENTH TRACK IS
        ;USED FOR PARITY.
        .MCALL ..V2..,.REGDEF
        ..V2..
        .REGDEF
        ERRWD=52
        .MCALL .CSIGEN,.READW,.PRINT,.EXIT,.CLOSE
RESET:  BIS #100000,@#167762
        BIS #2,@#167760
START:  .PRINT #ASK         ;DEVICE FILE?
        .CSIGEN #DSPACE,#DEXT;GET CS
        CLR INBLK           ;INITIALIZE BLK COUNT

READ:   .READW #LIST,#3,#BUF,#256.,INBLK;READ CHANNEL 3
        INC INBLK
        BCC GO
        TSTB @#ERRWD        ;TEST FOR EOF
        BEQ EOF
        MOV #INERR,R0
        .PRINT
        CLR R0
        JMP     RESET
GO:     MOV #BUF,R3         ;PREPARE TO READ OUT DATA
        MOV #OBUF,R4        ;R4 POINTS TO OUTPUT BUFF
FORMAT: CMP R3,#END         ;CHECK FOR END OF BUFF
        BEQ WRITE
        CMP R3,#END1        ;ONE ROW
        BEQ WRITE
RETURN: MOV (R3)+,R1        ;FIRST WD TO BE PROCESSED
                            SWAB R1
        ASHC #6,R0          ;PUSH BITS
        COM R0
        BIS #100000,R0      ;SET HIGH BIT
        MOV R0,(R4)+        ;MOVE BITS TO OUTBUF
        ASHC #6,R0
        COM R0
        BIS #100000,R0
        MOV R0,(R4)+        ;MOV BITS TO OUTBUF
        ASHC #4,R0
        MOV #2,SHIFT
        CMP R3,#END
        BEQ WRITE
        CMP R3,#END1
        BEQ WRITE
        MOV (R3)+,R1        ;GET ANOTHER WORD
                            SWAB R1
        ASHC #2,R0
```

```
        COM R0
        BIS #100000,R0
        MOV R0,(R4)+     ;MOVE BYTE TO OUTBUF
        ASHC #6,R0
        COM R0
        BIS #100000,R0
        MOV R0,(R4)+     ;MOVE BYTE TO OUTBUF
        ASHC #6,R0
        COM R0
        BIS #100000,R0
        MOV R0,(R4)+     ;MOVE BYTE TO OUTBUF
        ASHC #2,R0
        MOV #4,SHIFT
        CMP R3,#END
        BEQ WRITE
        CMP R3,#END1
        BEQ WRITE
        MOV (R3)+,R1
                         SWAB R1
        ASHC #4,R0
        COM R0
        BIS #100000,R0
        MOV R0,(R4)+
        ASHC #6,R0
        COM R0
        BIS #100000,R0
        MOV R0,(R4)+
        ASHC #6,R0
        COM R0
        BIS #100000,R0
        MOV R0,(R4)+
        BR FORMAT
WRITE:  ASHC SHIFT,R0
        COM R0
        BIS #100000,R0
        MOV R0,(R4)+     ;PUSH LEFT OVER BITS
        JSR PC ,OUT      ;WRITE OUT BITS
        MOV #OBUF,R4     ;RESET BUFF POINTER
        CMP R3,#END      ;REEAD ANOTHER
        BNE RETURN
        JMP READ         ;GET ANOTHER BLOCK
EOF:    CLR @#DATA       ;END OF FILE
        NOP
        MOV #100000,@#DATA
        .CLOSE #3
        .PRINT #FINIS
        JMP RESET
ASK:    .ASCIZ /   ENTER INPUT FILE  /
FINIS:  .ASCIZ /  ******FINISHED****** /
        .EVEN
        .WORD 0,0,0,0
DEXT:   .WORD 0,0,0,0
SHIFT:  .WORD 0
INBLK:  .WORD 0
LIST:   .BLKW 5
INERR:  .ASCIZ /   INPUT ERROR  /
```

```
        .EVEN
DSPACE: .BLKW 1000
BUF:
        .REPT 200
        .WORD 0
        .ENDR
END1:
        .REPT 200
        .WORD 0
        .ENDR
END:
OBUF:
        .REPT 1274
        .WORD 111111
        .ENDR
        .WORD 111111
        DATA=167762
        STATUS=167760
OUT:    MOV #OBUF,R1        ;PREPARE TO SEND OUT BITS
        MOV #530,R2         ;SIZE OF RECORD
AGAIN:  MOV (R1)+,@#DATA
        BIC #2,@#STATUS
RECORD: TSTB @#STATUS
        BNE RECORD
        BIS #2,@#STATUS
        MOV (R1)+,@#DATA
        BIC #2,@#STATUS
        SOB R2,RECORD
        BIS #2,@#STATUS
        RTS PC
        .END RESET
```

```
        ;PROGRAM MAG512          AUG 10,1977          MACHUCA
        ;COPIES PICTURE FILES CONTAINING CONSECUTIVE ROWS OF 512
        ;PIXELS FROM FLOPPY DISKS ONTO 7-TRACK MAGNETIC TAPE.  THE
        ;8-BIT PIXELS ARE STACKED ONTO THE CONSECUTIVE LINES
        ;OF THE TAPE AS DESCRIBED IN THE PROGRAM MAG256.
         MCALL   V2...  REGDEF
          V2..
         REGDEF
        ERRWD=52
         MCALL  CSIGEN,.READW, PRINT,.EXIT, CLOSE
RESET   BIS     #100000,@#167762
        BIS     #2,@#167760
START    PRINT #ASK      ;DEVICE FILE?
         CSIGEN #DSPACE,#DEXT,GET CS
        CLR INBLK       ;INITIALIZE BLK COUNT

READ     READW #LIST,#3,#BUF,#256,INBLK;READ CHANNEL 3
        INC INBLK
        BCC GO
        TSTB @#ERRWD    ;TEST FOR EOF
        BEQ EOF
        MOV #INERR,R0
         PRINT
        CLR R0
        JMP     RESET
GO      MOV #BUF,R3     ;PREPARE TO READ OUT DATA
        MOV #OBUF,R4    ;R4 POINTS TO OUTPUT BUFF
FORMAT  CMP R3,#END     ;CHECK FOR END OF BUFF
        BEQ WRITE
RETURN  MOV (R3)+,R1    ;FIRST WD TO BE PROCESSED
                        SWAB R1
        ASHC #6,R0      ;PUSH BITS
        COM R0
        BIS #100000,R0  ;SET HIGH BIT
        MOV R0,(R4)+    ;MOVE BITS TO OUTBUF
        ASHC #6,R0
        COM R0
        BIS #100000,R0
        MOV R0,(R4)+    ;MOV BITS TO OUTBUF
        ASHC #4,R0
        MOV #2,SHIFT
        CMP R3,#END
        BEQ WRITE
        MOV (R3)+,R1    ;GET ANOTHER WORD
                        SWAB R1
        ASHC #2,R0
        COM R0
        BIS #100000,R0
        MOV R0,(R4)+    ;MOVE BYTE TO OUTBUF
        ASHC #6,R0
        COM R0
        BIS #100000,R0
        MOV R0,(R4)+    ;MOVE BYTE TO OUTBUF
        ASHC #6,R0
        COM R0
```

```
        BIS #100000,R0
        MOV R0,(R4)+      ;MOVE BYTE TO OUTBUF
        ASHC #2,R0
        MOV #4,SHIFT
        CMP R3,#END
        BEQ WRITE
        MOV (R3)+,R1
                          SWAB R1
        ASHC #4,R0
        COM R0
        BIS #100000,R0
        MOV R0,(R4)+
        ASHC #6,R0
        COM R0
        BIS #100000,R0
        MOV R0,(R4)+
        ASHC #6,R0
        COM R0
        BIS #100000,R0
        MOV R0,(R4)+
        BR FORMAT
WRITE   ASHC SHIFT,R0
        COM R0
        BIS #100000,R0
        MOV R0,(R4)+      ;PUSH LEFT OVER BITS
        JSR PC ,OUT       ;WRITE OUT BITS
        MOV #OBUF,R4      ;RESET BUFF POINTER
        CMP R3,#END       ;REEAD ANOTHER
        BNE RETURN
        JMP READ          ;GET ANOTHER BLOCK
EOF     CLR @#DATA        ;END OF FILE
        NOP
        MOV #100000,@#DATA
         CLOSE #3
         PRINT #FINIS
        JMP     RESET
ASK      ASCIZ /   ENTER INPUT FILE  /
FINIS    ASCIZ /  ******FINISHED****** /
         EVEN
         WORD 0,0,0,0
DEXT     WORD 0,0,0,0
SHIFT    WORD 0
INBLK    WORD 0
LIST     BLKW 5
INERR    ASCIZ /   INPUT ERROR  /
         EVEN
DSPACE   BLKW 1000
BUF
         REPT 200
         WORD 0
         ENDR
         REPT 200
         WORD 0
         ENDR
END
OBUF
```

```
          REPT 1274
          WORD 111111
          ENDR
          WORD 111111
         DATA=167762
         STATUS=167760
OUT      MOV #OBUF,R1      ;PREPARE TO SEND OUT BITS
         MOV #1260,R2      ;SIZE OF RECORD
AGAIN   MOV (R1)+,@#DATA
         BIC #2,@#STATUS
RECORD   TSTB @#STATUS
         BNE RECORD
         BIS #2,@#STATUS
         MOV (R1)+,@#DATA
         BIC #2,@#STATUS
         SOB R2,RECORD
         BIS #2,@#STATUS
         RTS PC
          END RESET
```

PDP-11 FORTRAN SYSTEM PROGRAMS AND MAJOR SUBROUTINES

```
      PROGRAM HISTP
C     ************************************************************
C     *                                                          *
C     *   HISTP COMPUTES THE HISTOGRAM OF AN ENTIRE              *
C     *   PICTURE FILE.  THE NUMBER OF COLUMNS AND THE           *
C     *   NUMBER OF GREY LEVELS IN A BIN ARE SPECIFIED           *
C     *   BY THE USER.                                           *
C     *                                                          *
C     *//////////////////////////////////////////////////////////*
C     *                                                          *
C     *   SUBROUTINES REQUIRED:                                  *
C     *       FROM IOOP-WRITEF                                   *
C     *       FROM GRAPH-PHIST,GRID,AXIS,FRAME,                  *
C     *                  INIT,TICX                               *
C     *       SEVERAL SYSLIB SUBROUTINES                         *
C     *                                                          *
C     ************************************************************
C
      LOGICAL*1 A(30720),ID(12),AG(1)
      INTEGER*2 AH(256),X(17),Y(17)
      COMMON /GRAP/LU,IX,IY,ISX,ISY,AG
      LU=6
      CALL START(LU)
10    CALL WRITEF(LU,ICH,ID)
      WRITE(5,1)
1     FORMAT(1X,'INPUT YOUR CHOICE FOR THE # OF
     1 GREY LEVELS IN A BIN  ',$)
      READ(5,2)IB
2     FORMAT(I4)
      NB=256/IB
      WRITE(5,4)
4     FORMAT(1X,'ENTER NUMBER OF COLUMNS IN THE INPUT PICTURE ')
      READ(5,2)NCC
      NR=256*120/NCC
      IFF=240/NR
      IS=1
      DO 15 I=1,NB
      AH(I)=0
15    CONTINUE
      DO 40 I=1,IFF
      IBB=(I-1)*60
      JS=IREADW(15360,A,IBB,ICH)
      DO 20 KK=1,30720
      L=(A(KK) AND 255)/IB+1
      AH(L)=AH(L)+1
C
C IF THE COUNT GETS TOO LARGE SCALE
C THE HIST BY 2
C
      IF(AH(L).GT.30000)GO TO 60
20    CONTINUE
40    CONTINUE
      CALL INIT(5)
      WRITE(5,3)ID,IB,IS
3     FORMAT(1X,'HISTOGRAM OF THE FILE - ',3A1,': ',6A1
     1,'. ',3A1/1X,'THE NUMBER OF GREY LEVELS IN A BIN IS ',
```

```
     1I4/1X,'THE ORDINATE WAS  SCALED ',I1,'/1')
      ISX=900
      CALL PHIST(AH,NB,MAX)
      DO 50 I=1,17
      X(I)=256-(17-I)*16
      Y(I)=MAX*FLOAT(I-1)/16.0
50    CONTINUE
      CALL FRAME
      CALL GRID(16)
      CALL AXIS(X,Y,17,17)
      CALL TICX(20,16,4)
      READ(5,2)L
      DO 55 I=1,NB
      WRITE(5,5)(I-1)*IB,AH(I)
5     FORMAT(1X,2(I6,1X))
55    CONTINUE
      CALL CLOSE(LU)
      CALL CLOSEC(ICH)
      CALL IFREEC(ICH)
      GO TO 10
60    DO 65 KKK=1,NB
      AH(KKK)=AH(KKK)/2
65    CONTINUE
      IS=IS*2
      GO TO 20
      END
```

```
      SUBROUTINE PHIST(IV,N,MAX)
C
C     ************************************************************
C     *                                                          *
C     *   PHIST PLOTS A HISTOGRAM OF THE VALUES IN ARRAY IV.     *
C     *   THE HISTOGRAM IS NORMALIZED TO FIT INSIDE THE          *
C     *   WINDOW AS SET FORTH BY THE COMMON VARIABLES.  THE      *
C     *   BARS WIDTHS ARE EQUAL TO THE SIZE OF THE GRAPHIC       *
C     *   WINDOW IN THE X DIRECTION (ISX) DIVIDED BY THE         *
C     *   NUMBER OF BINS.  THE MAXIMUM COUNT IN THE ARRAY        *
C     *   IV IS PASSED BACK THROUGH THE PARAMETER MAX.           *
C     *                                                          *
C     *//////////////////////////////////////////////////////////*
C     *                                                          *
C     *   PARAMETERS                                             *
C     *        IV=THE INTEGER*2 ARRAY CONTAINING THE N COUNTS    *
C     *        N =THE NUMBER OF COUNTS IN IV                     *
C     *       MAX=THE MAXIMUM COUNT FOUND IN IV                  *
C     *                                                          *
C     *   LIMITS:                                                *
C     *        THE COUNTS IN IV MUST NOT EXCEED 30,000 AS        *
C     *        THEN THEY BEGIN TO LOOK LIKE NEGATIVE NUMBERS     *
C     *                                                          *
C     *//////////////////////////////////////////////////////////*
C     *                                                          *
C     *   SUBROUTINES REQUIRED                                   *
C     *        LX,LY,CKHX,LOADB                                  *
C     *                                                          *
C     ************************************************************
C
      LOGICAL*1 A(1),HY,LAY,LAX,HX
      INTEGER*2 IV(N)
      COMMON /GRAP/LU,IX,IY,ISX,ISY,A
C
      JW=ISX/N
      NWM=JW-1
C
C     FIND THE MAX COUNT
C
      MAX=IV(1)
      DO 10 I=2,N
      IF(MAX.LT.IV(I))MAX=IV(I)
   10 CONTINUE
C
C     LOAD THE OUTBUFFER
C
      A(1)=29
      CALL LOADB(IX,IY,A,2)
      HY=A(2)
      LAY=A(3)
      HX=A(4)
      LAX=A(5)
      IN=6
      DO 40 I=1,N
C
      KY=FLOAT(IV(I))/FLOAT(MAX)*ISY+IY
      KX=IX+(I-1)*JW
```

```
      IF(KY.EQ.IY)GO TO 30
      CALL LY(KY,A,IN)
      A(IN+2)=LAX
      A(IN+3)=A(IN+1)
      IN=IN+4
      CALL CKHX(KX+NWM,A,IN,HX)
C
      LAX=A(IN-1)
      A(IN)=HY
      A(IN+1)=LAY
      A(IN+2)=LAX
      A(IN+3)=LAY
      IN=IN+4
 20   CALL CKHX(KX+JW,A,IN,HX)
C
      LAX=A(IN-1)
      GO TO 40
 30   A(IN)=LAY
      IN=IN+1
      GO TO 20
 40   CONTINUE
C
      WRITE(LU,50)(A(I),I=1,IN-1)
 50   FORMAT('+',$,124A1)
      RETURN
      END
```

```
      PROGRAM WRP256
C
C     ************************************************************
C     *                                                          *
C     *   WRP256 WRITES A GREY LEVEL PLOT TO THE OUTPUT          *
C     *   FILE CONNECTED TO LOGICAL UNIT LU IN WRITEF.           *
C     *   THE INPUT FILE MUST BE ROW STRUCTURED AND OF           *
C     *   256 COLUMNS.                                           *
C     *                                                          *
C     *   THE WINDOW OF DATA READ FROM THE FILE IS PLOTTED       *
C     *   UPSIDE DOWN WITH THE LOWER LEFT HAND CORNER            *
C     *   SPECIFIED BY USER INPUT FROM THE TTY.                  *
C     *                                                          *
C     *   PICTURE FILES OBTAINED FROM DIG2F1 OR DIG2F2           *
C     *   MAY BE DISPLAYED WITH A NEARLY NATURAL ASPECT          *
C     *   RATIO OF 240/256 USING WRP256.  PICTURE FILES          *
C     *   OBTAINED FROM DIG5F1 OF DIG5F2 WILL BE DISPLAYED       *
C     *   WITH AN ASPECT RATIO OF 240/512 AND WILL THERE-        *
C     *   FORE APPEAR TO BE COMPRESSED IN THE Y DIRECTION        *
C     *   AND STRETCHED IN THE X DIRECTION.  THE DATA            *
C     *   FROM THE DIGITIZER MUST BE TRANSPOSED USING            *
C     *   TR256 BEFORE IT CAN BE DISPLAYED USING WRP256          *
C     *                                                          *
C     ************************************************************
C     *                                                          *
C     *   SUBROUTINES REQUIRED:                                  *
C     *               FROM IOOP-WRITEF                           *
C     *               FROM GRAPH-GSX                             *
C     *               SEVERAL SYSLIB SUBROUTINE                  *
C     *                                                          *
C     ************************************************************
C
      LOGICAL*1 A(256,96),IZ(256),DB(12)
   10 LU=6
      CALL WRITEF(LU,ICH,DB)
      CALL START(LU)
      TYPE 11
   11 FORMAT(1X,'** MAX SUM BEGINNING ROW PLUS SPREAD=240 **')
      GO TO 19
   17 TYPE 18
   18 FORMAT(1X,'YOUR INPUT SPREAD FOR ROWS EXCEEDS THE MAX
     1 LIMIT OF 239')
   19 WRITE(5,20)
   20 FORMAT(1X,'INPUT THE BEGINNING ROW AND SPREAD- ')
      READ(5,50)IR,IRS
      IF(IRS.GT.239)GO TO 17
      TYPE 12
   12 FORMAT(1X,'** MAX SUM BEGINNING COLUMN PLUS SPREAD=256 **')
      GO TO 23
   21 TYPE 22
   22 FORMAT(1X,'YOUR INPUT SPREAD FOR COLUMNS EXCEEDS THE
     1MAX. LIMIT OF 255')
   23 WRITE(5,25)
   25 FORMAT(1X,'INPUT THE BEGINNING COLUMN AND SPREAD- ')
      READ(5,50)IC,ICS
      IF(ICS.GT.255)GO TO 21
```

```
      WRITE(5,30)
30    FORMAT(1X,'ENTER THE X AND Y OF THE LOWER LEFT HAND
     1 POINT OF THE PLOT- ',$)
      READ(5,50)LLX,LLY
      WRITE(5,40)
40    FORMAT(1X,'INPUT YOUR VALUE FOR BLACK AND
     1 CONTRAST SPREAD')
      READ(5,50)IB,IS
50    FORMAT(4I4)
C
C     RESET THE SCREEN
C
      CALL INIT(LU)
      IFB=(IR-1)/2
      IFF=IRS/(97)+1
      IWC=96*128
      IF(IRS.LT.96)IWC=IRS*128
      IBB=IFB
      ICHS=IC+ICS-1
      S=IS
      DO 80 K=1,IFF
      JS=IREADW(IWC,A,IBB,ICH)
      JJ=96
      IF(K.EQ.IFF)JJ=IWC/128
C
      DO 70 J=1,JJ
      DO 60 I=1,ICHS
      II=ICHS-I+1
      KK=A(II,J).AND.255
      FK=KK-IB
      IF(FK.LT.0)FK=0
      IF(FK.GT.S)FK=S
      ZZ=16.-(16.0*FK/S)
      IZZ=ZZ
      IF((ZZ-IZZ).GE..5)IZZ=IZZ+1
      IZ(I)=IZZ
60    CONTINUE
C
      CALL GSX(LLX,J+LLY-1+(K-1)*96,IZ,ICS)
70    CONTINUE
      IBB=48+IBB
      IF(K.EQ.(IFF-1))IWC=(IRS-K*96)*128
80    CONTINUE
      CALL CLOSE(LU)
      CALL CLOSEC(ICH)
      CALL IFREEC(ICH)
      READ(5,50)IS
      GO TO 10
      END
```

```
      PROGRAM WRP512
C
C     ************************************************************
C     *                                                          *
C     *   WRP512 WRITES A GREY LEVEL PLOT ON LOGICAL UNIT        *
C     *   NUMBER LU OF THE ROWS AND COLUMNS OF THE INPUT         *
C     *   FILE SPECIFIED BY THE USER.  THE INPUT AND OUT-        *
C     *   PUT FILE ARE ENTERED IN STANDARD CSI, THE OUT-         *
C     *   PUT BEING CONNECTED TO LU.  THE PICTURE IS             *
C     *   PRINTED UPSIDE DOWN WITH THE LOWER LEFT CORNER         *
C     *   ENTERED BY THE USER                                    *
C     *                                                          *
C     *   THE INPUT PICTURE FILE MUST CONTAIN SEQUENTIAL         *
C     *   ROWS WITH 512 PIXELS PER ROW.  THE PROGRAM COM-        *
C     *   PRESSES EACH ROW INTO 256 PIXELS BY TAKING THE         *
C     *   ROW ELEMENTS TWO AT A TIME, COMPUTING THE AVERAGE      *
C     *   INTENSITY OF EACH PAIR OF ELEMENTS, AND THEN           *
C     *   PLACING THE RESULTING AVERAGE INTENSITIES IN A         *
C     *   256 ELEMENT ROW OF MATRIX 'A'.  THE COLUMN SPREAD      *
C     *   REQUESTED BY THE PROGRAM REFERS TO THE COMPRESSED      *
C     *   MATRIX 'A' AND THEREFORE CANNOT EXCEED 256             *
C     *   ELEMENTS.  THE RESULTING PICTURE IS DISPLAYED          *
C     *   WITH A NEARLY NATURAL ASPECT RATIO OF 240/256          *
C     *   WHEN THE INPUT PICTURE FILES ARE OBTAINED FROM         *
C     *   DIG5F1 OR DIG5F2.  THE DATA FROM THE DIGITIZER         *
C     *   MUST BE TRANSPOSED USING TR512 BEFORE IT CAN BE        *
C     *   DISPLAYED USING WRP512                                 *
C     *                                                          *
C     ************************************************************
C     *                                                          *
C     *   SUBROUTINES REQUIRED:                                  *
C     *              FROM IOOP-WRITEF                            *
C     *              FROM GRAPH-GSX                              *
C     *              SEVERAL SYSLIB SUBROUTINE                   *
C     *                                                          *
C     ************************************************************
C
      LOGICAL*1 A(512,64),IZ(256),DB(12)
      LU=6
      CALL START(LU)
   10 CALL WRITEF(LU,ICH,DB)
      TYPE 11
   11 FORMAT(1X,'** MAX. SUM BEGINNING ROW PLUS SPREAD=240 **')
      GO TO 19
   17 TYPE 18
   18 FORMAT(1X,'YOUR INPUT SPREAD FOR ROWS EXCEEDS THE MAX
     1 LIMIT OF 239')
   19 WRITE(5,20)
   20 FORMAT(1X,'INPUT THE BEGINNING ROW AND SPREAD- ')
      READ(5,50)IR,IRS
      IF(IRS.GT.239)GO TO 17
      TYPE 12
   12 FORMAT(1X,'** MAX. SUM BEGINNING COLUMN PLUS SPREAD=256 **')
      GO TO 23
   21 TYPE 22
   22 FORMAT(1X,'YOUR INPUT SPREAD FOR COLUMNS EXCEEDS THE
```

```
     1 MAX. LIMIT OF 255')
  23    WRITE(5,25)
  25    FORMAT(1X,'INPUT THE BEGINNING COLUMN AND SPREAD- ')
        READ(5,50)IC,ICS
        IF (ICS.GT.255)GO TO 21
        WRITE(5,30)
  30    FORMAT(1X,'ENTER THE X AND Y OF THE LOWER LEFT HAND
     1 POINT OF THE PLOT- '$)
        READ(5,50)LLX,LLY
        WRITE(5,40)
  40    FORMAT(1X,'INPUT YOUR VALUE FOR BLACK AND)
     1 CONTRAST SPREAD')
        READ(5,50)IB,IS
  50    FORMAT(4I4)
C
C       RESET THE SCREEN
C
        CALL INIT(LU)
        IBB=IR-1
        IFF=IRS/65+1
        IWC=64*256
        IF (IRS.LT.64)IWC=IRS*256
        S=IS
        ICHS=IC+ICS-1
        JJ=64
        DO 100 K=1,IFF
        JS=IREADW(IWC,A,IBB,ICH)
        IF(K.EQ.IFF)JJ=IWC/256
C
        DO 70 II=1,ICHS
        III=II*2-1
        DO 60 JJJ=1,JJ
        A(II,JJJ)=((A(III,JJJ).AND.255)+(A(III+1,JJJ).AND.255))/2
  60    CONTINUE
  70    CONTINUE
        DO 90 J=1,JJ
        DO 80 I=1,ICS
        II=ICS-I+1
        KK=A(II,J).AND.255
        FK=KK-IB
        IF(FK.LT.0)FK=0
        IF(FK.GT.S)FK=S
        ZZ=16.-(16.0*FK/S)
        IZZ=ZZ
        IF((ZZ-IZZ).GE..5)IZZ=IZZ+1
        IZ(I)=IZZ
  80    CONTINUE
C
        CALL GSX(LLX,J+LLY-1+(K-1)*64,IZ,ICS)
  90    CONTINUE
        IBB=64+IBB
        IF(K.EQ.(IFF-1))IWC=(IRS-K*64)*256
 100    CONTINUE
        CALL CLOSEC(ICH)
        CALL IFREEC(ICH)
```

```
CALL CLOSE(LU)
READ(5,50)IS
GO TO 10
END
```

```
      PROGRAM DHIST
C
C     ***************************************************************
C     *                                                             *
C     *   DHIST COMPUTES THE ONE (1) DIMENSIONAL HISTOGRAM          *
C     *   OF A SPECIFIED WINDOW IN THE INPUT FILE.  THE             *
C     *   HISTOGRAM IS PLOTTED OUT WITH USER SPECIFIED              *
C     *   BIN SIZE.  THE GREY LEVEL PICTURE DISPLAYS ANY            *
C     *   PORTION OF A 256 OR 512 COLUMN PICTURE FILE.              *
C     *   THE DIMENSIONS OF THE DISPLAYED WINDOW ARE SEL-           *
C     *   ECTED BY THE USER UP TO A MAXIMUM WINDOW SIZE             *
C     *   OF 100 X 100 PIXELS.                                      *
C     *                                                             *
C     *   IN ADDITION, DHIST WILL DISPLAY COMPLETE PICTURE          *
C     *   FILES AS LARGE AS 64 X 64 PIXELS IF THE TRIMMED           *
C     *   PICTURE FILE DIMENSIONS ARE 2**N X 2**M WHERE N           *
C     *   AND M CAN TAKE ON ANY VALUES FROM 1 TO 6.  IN             *
C     *   THIS CASE THE EXACT PICTURE FILE DIMENSIONS               *
C     *   MUST BE ENTERED AS THE DIMENSIONS OF THE WINDOW           *
C     *   TO BE DISPLAYED.                                          *
C     *                                                             *
C     *************************************************************
C     *                                                             *
C     *   SUBROUTINES REQUIRED:                                     *
C     *               FROM IOOP-WRITEF,READR                        *
C     *               FROM GRAPH-ALL OF THEM                        *
C     *               SUBROUTINE HIST                               *
C     *               SEVERAL SYSLIB SUBROUTINES                    *
C     *                                                             *
C     ***************************************************************
C
      COMMON//L
      LOGICAL*1 DB(12)
      INTEGER*2 A(256),IX(20),IY(20)
      L=1
      LU=6
   10 CALL WRITEF(LU,ICH,DB)
      CALL START(LU)
   20 WRITE(5,30)
   30 FORMAT(1X,'INPUT THE SIZE OF THE HISTOGRAM BINS')
      READ(5,40)NB
   40 FORMAT(I4)
C
      CALL HIST(LU,ICH,A,DB)
      IB=256/NB
      DO 60 I=1,IB
      IA=0
      IN=(I-1)*NB
      DO 50 J=1,NB
      IA=IA+A(IN+J)
   50 CONTINUE
C
      A(I)=IA
   60 CONTINUE
      CALL PHIST(A,IB,MAX,NB)
      DO 70 I=1,17
```

```
      IX(I)=256-(17-I)*16
      IY(I)=MAX*FLOAT(I-1)/16.0
   70 CONTINUE
      CALL FRAME()
      CALL GRID(16)
      CALL AXIS(IX,IY,17,17)
      CALL TICX(20,16,4)
C
      READ(5,100)L
      CALL INIT(LU)
      WRITE(5,80)
      WRITE(5,90)((I-1)*NB,A(I),I=1,IB)
   80 FORMAT(1X,'THE COUNTS FOR THE GREY LEVEL HISTOGRAM
     1 WERE AS FOLLOWS:'/1X,5X,'GREY LEVEL',4X,'COUNT')
   90 FORMAT(6X,I3,8X,I4)
  100 FORMAT(I1)
      WRITE(5,110)
  110 FORMAT(1X,'WOULD YOU LIKE A NEW INPUT FILE?(YES=1,NO=0)')
      READ(5,100)L
      IF((L).EQ.1)GO TO 120
      GO TO 20
  120 CALL CLOSE(LU)
      GO TO 10
      END
```

```
      SUBROUTINE HIST(LU, ICH, A, DB)
C
C     ***************************************************************
C     *                                                             *
C     *  HIST IS A PROGRAM DESIGNED TO BE CALLED BY DHIST           *
C     *  ALMOST EXCLUSIVELY.  HIST READS IN A WINDOW                *
C     *  OF DATA FROM A ROW STRUCTURED PICTURE FILE                 *
C     *  ACCORDING TO USER ENTERED ROW AND COLUMN LIMITS.           *
C     *  OPTIONS ARE INCLUDED TO ALLOW THE GREY LEVEL               *
C     *  PICTURE OF THE WINDOW BE DRAWN OUT IN THE UPPER            *
C     *  RIGHT HAND CORNER OF THE SCREEN, AND ALSO TO               *
C     *  SAVE DATA PREVIOUSLY READ.                                 *
C     *                                                             *
C     *///////////////////////////////////////////////////////////*
C     *                                                             *
C     *  PARAMETERS:                                                *
C     *      LU=THE LOGICAL UNIT NUMBER TO WRITE THE                *
C     *         GRAPHICS OUT TO                                     *
C     *      ICH=THE CHANNEL NUMBER TO READ THE DATA FROM           *
C     *      A=THE LOGICAL*1 ARRAY TO READ DATA INTO                *
C     *      DB=THE LOGICAL*1 12 BYTE VECTOR WHICH CONTAINS         *
C     *         THE ASCII FOR THE READ FILE                         *
C     *                                                             *
C     *///////////////////////////////////////////////////////////*
C     *                                                             *
C     *  LIMITS:                                                    *
C     *      THE WINDOW OF DATA READ CAN BE AT MOST                 *
C     *      100 X 100                                              *
C     *                                                             *
C     *///////////////////////////////////////////////////////////*
C     *                                                             *
C     *  SUBROUTINES REQUIRED:                                      *
C     *      FROM IOOP - READR                                      *
C     *      FROM GRAPH - GRA, INIT                                 *
C     *      SEVERAL SYSLIB SUBROUTINES                             *
C     *                                                             *
C     ***************************************************************
C
      COMMON//L
      INTEGER*2 A(256)
      LOGICAL*1 AS(100, 100), DB(12), B(40)
      DO 10 I=1, 256
      A(I)=0
   10 CONTINUE
C
      IF((L).EQ.1)GO TO 125
      WRITE (5, 20)
   20 FORMAT(1X, 'WOULD YOU LIKE TO SAVE PREVIOUS DATA?(YES=1, NO=0)')
      READ (5, 40)ITT
      IF(ITT.EQ.1)GO TO 60
  125 TYPE 21
   21 FORMAT(1X, 'SPECIFY PORTION OF INPUT PICTURE TO BE DISPLAYED')
      TYPE 22
   22 FORMAT(15X, '*** MAX. SPREAD 100 X 100 ***')
   25 WRITE(5, 30)
   30 FORMAT(1X, 'INPUT THE BEGINNING ROW AND SPREAD. ')
```

```
      READ (5,40)II,IR
      WRITE (5,35)
 35   FORMAT(1X,'INPUT THE BEGINNING COLUMN AND SPREAD ')
      READ (5,40)KK,IC
 40   FORMAT(4I4)
C
      JJ=IR+II-1
      LL=IC+KK-1
      WRITE(5,50)
 50   FORMAT(1X,'ENTER NUMBER OF COLUMNS IN THE INPUT PICTURE FILE')
      READ(5,40)NC
      CALL READR(II,JJ,KK,LL,100,AS,ICH,NC)
 60   WRITE(5,70)
 70   FORMAT(1X,'WOULD YOU CARE TO TITLE YOUR HISTOGRAM?')
      READ(5,80)(B(I),I=1,40)
 80   FORMAT(40A1)
C
      WRITE(5,90)
 90   FORMAT(1X,'WOULD YOU LIKE TO SEE THE GRAY LEVELS?(YES=1,NO=0) ')
      READ(5,40)ITT
      IF(ITT EQ 1)GO TO 160
      CALL INIT(LU)
 100  WRITE (LU,140)(DB(I),I=4,12)
      WRITE(LU,150)II,JJ,KK,LL
      WRITE(5,110)(B(I),I=1,40)
 110  FORMAT(//21X,40A1)
C
      DO 130 I=1,IR
      DO 120 J=1,IC
      K=(AS(I,J) AND 255)+1
      A(K)=A(K)+1
 120  CONTINUE
 130  CONTINUE
 140  FORMAT(1X,'THE INPUT FILE WAS    ',6A1,' '3A1)
 150  FORMAT(1X,'ROWS ',I3,' TO ',I3/1X,'COLS ',I3,' TO ',I3)
C
      RETURN
 160  CALL GRA(IR,IC,100,AS)
      READ(5,40)ITT
      GO TO 100
      END
```

```
      PROGRAM TR256
C
C     ************************************************************
C     *                                                          *
C     *    TRANSPOSES A 240 BY 256 ARRAY OF BYTES STORED         *
C     *    IN SEQUENTIAL COLUMNS TO A 256 BY 240 ARRAY           *
C     *    STORED IN ROWS.                                       *
C     *                                                          *
C     ************************************************************
C     *                                                          *
C     *    SUBROUTINES REQUIRED:                                 *
C     *         FROM IOOP-TRANS2, RENTER, WENTRN, AREAD          *
C     *         SEVERAL SYSLIB SUBROUTINES                       *
C     *                                                          *
C     ************************************************************
C
      LOGICAL*1 A(30,30)
2     CALL TRANS2()
      ICH=IGETC()
      CALL RENTER(ICH)
      CALL AREAD(1,30,1,30,30,A,ICH)
      DO 10 I=1,30
      WRITE(5,1)(A(I,J),J=1,30)
1     FORMAT(1X,30(I3,1X))
10    CONTINUE
      CALL CLOSEC(ICH)
      CALL IFREEC(ICH)
      GO TO 2
      END
```

```
      SUBROUTINE TRANS2()
C
C       ************************************************************
C       *                                                          *
C       *   TRANSPOSES A 240 BY 256 ARRAY OF BYTES STORED          *
C       *   IN SEQUENTIAL COLUMNS TO A 256 BY 240 ARRAY            *
C       *   STORED IN ROWS.  THIS SUBROUTINE IS CALLED             *
C       *   BY TR256                                               *
C       *                                                          *
C       ************************************************************
C       *                                                          *
C       *   SUBROUTINES REQUIRED:                                  *
C       *        FROM IOOP-RENTER, WENTRN                          *
C       *        SEVERAL SYSLIB SUBROUTINES                        *
C       *                                                          *
C       ************************************************************
C
      INTEGER*2 BUFF(256)
      LOGICAL*1 A(80,256),AS(512)
      EQUIVALENCE (BUFF(1),AS(1))
C
      ICHR=IGETC()
      WRITE(5,1)
 1    FORMAT(1X,'ENTER THE FILE TO BE TRANSPOSED ')
      CALL RENTER(ICHR)
      WRITE(5,2)
 2    FORMAT(1X,'ENTER OUTPUT FILE FOR THE NEW TRANSPOSED PICTURE')
      ICHW=IGETC()
      CALL WENTRN(ICHW,120)
C
C TRANSPOSE THE ARRAY IN BLOCKS 40 ROWS BY 256 COLS
C
      DO 30 I=1,3
C
C SET BEGINNING ROW COUNTER
C
      IR=(I-1)*80+1
      IRR=IR+79
      CALL AREAD(IR,IRR,1,256,80,A,ICHR)
C
C FILL UP THE WRITE BUFFER WITH 2 ROWS
C
      II=1
      DO 20 IB=1,40
      IBB=IB-1
      DO 10 J=1,256
      AS(J)=A(II,J)
      AS(J+256)=A(II+1,J)
 10   CONTINUE
C
C WRITE OUT THE BUFFER OF TWO ROWS TO THE TRANS FILE
C
      II=II+2
      JB=IBB+(I-1)*40
      JS=IWRITW(256,BUFF,JB,ICHW)
```

```
20    CONTINUE
30    CONTINUE
      CALL CLOSEC(ICHR)
      CALL CLOSEC(ICHW)
      CALL IFREEC(ICHR)
      CALL IFREEC(ICHW)
      RETURN
      END
```

```
      PROGRAM TR512
C
C     ***************************************************************
C     *                                                             *
C     *   TRANSPOSES A 240 BY 512 ARRAY OF BYTES STORED             *
C     *   IN SEQUENTIAL COLUMNS TO A 512 BY 240 ARRAY               *
C     *   STORED IN ROWS                                            *
C     *                                                             *
C     ///////////////////////////////////////////////////////////////*
C     *                                                             *
C     *   SUBROUTINES REQUIRED:                                     *
C     *        FROM IOOP--TRANS5, RENTER, WENTRN, AREAD             *
C     *        SEVERAL SYSLIB SUBROUTINES                           *
C     *                                                             *
C     ***************************************************************
C
      LOGICAL*1 A(30,30)
2     CALL TRANS5()
      ICH=IGETC()
      CALL RENTER(ICH)
      CALL AREAD(1,30,1,30,30,A,ICH)
      DO 10 I=1,30
      WRITE(5,1)(A(I,J),J=1,30)
1     FORMAT(1X,30(I3,1X))
10    CONTINUE
      CALL CLOSEC(ICH)
      CALL IFREEC(ICH)
      GO TO 2
      END
```

```
      SUBROUTINE TRANS5()
C
C     ***************************************************************
C     *                                                             *
C     *   TRANSPOSES A 240 BY 512 ARRAY OF BYTES STORED             *
C     *   IN SEQUENTIAL COLUMNS TO A 512 BY 240 ARRAY               *
C     *   STORED IN ROWS.  THIS SUBROUTINE IS CALLED                *
C     *   BY TR512                                                  *
C     *                                                             *
C     *///////////////////////////////////////////////////////////////*
C     *                                                             *
C     *   SUBROUTINES REQUIRED:                                     *
C     *        FROM IOOP-RENTER, WENTRN                             *
C     *        SEVERAL SYSLIB SUBROUTINES                           *
C     *                                                             *
C     ***************************************************************
C
      INTEGER*2 BUFF(256)
      LOGICAL*1 A(60,512),AS(512)
      EQUIVALENCE (BUFF(1),AS(1))
C
      ICHR=IGETC()
      WRITE(5,1)
    1 FORMAT(1X,'ENTER THE FILE TO BE TRANSPOSED')
      CALL RENTER(ICHR)
      WRITE(5,2)
    2 FORMAT(1X,'ENTER OUTPUT FILE FOR THE NEW TRANSPOSED
     1 PICTURE')
      ICHW=IGETC()
      CALL WENTRN(ICHW,240)
C
C     TRANSPOSE THE ARRAY IN BLOCKS 60 ROWS BY 512 COLUMNS
C
      DO 30 I=1,4
C
C     SET BEGINNING ROW COUNTER
C
      IR=(I-1)*60+1
      IRR=IR+59
      CALL AREAD(IR,IRR,1,512,60,A,ICHR)
C
C     FILL UP THE WRITE BUFFER WITH 1 ROW
C
      DO 20 IB=1,60
      IBB=IB-1
      DO 10 J=1,512
      AS(J)=A(IB,J)
   10 CONTINUE
C
C     WRITE OUT THE BUFFER OF 1 ROW TO THE TRANS FILE
C
      JB=IBB+(I-1)*60
      JS=IWRITW(256,BUFF,JB,ICHW)
   20 CONTINUE
   30 CONTINUE
      CALL CLOSEC(ICHR)
```

```
CALL CLOSEC(ICHW)
CALL IFREEC(ICHR)
CALL IFREEC(ICHW)
RETURN
END
```

```
C
C       ***************************************************************
C       *
C       *     TRIM ACCEPTS PICTURE FILES OF ANY SIZE UP TO 240 BY
C       *     512   IT SELECTS A SPECIFIED PORTION OF THE INPUT PICTURE
C       *     FILE AND OUTPUTS THE TRIMMED PICTURE TO AN OUTPUT PICTURE
C       *     FILE.
C       *
C       *///////////////////////////////////////////////////////////////
C       *
C       *     SUBROUTINES REQUIRED
C       *           ASK
C       *           SEVERAL SYSLIB SUBROUTINES
C       *
C       ***************************************************************
C
        PROGRAM TRIM
        LOGICAL*1 A(600)
        INTEGER COLUM,LENGHT,ROW,WIDTH,POINT1,POINT2
        INTEGER PICX,X(300)
        CALL ASK
C
        TYPE 10
   10   FORMAT('   LENGTH OF ROWS FROM INPUT PICTURE FILE-    ',$)
        ACCEPT 60,PICX
        TYPE 20
   20   FORMAT('   BEGINNING COLUMN OF INPUT PICTURE FILE-    ',$)
        ACCEPT 60,COLUM
        TYPE 30
   30   FORMAT('   BEGINNING ROW OF INPUT PICTURE FILE-       ',$)
        ACCEPT 60,ROW
        TYPE 40
   40   FORMAT('   NUMBER OF COLUMNS FOR TRIM PICTURE-        ',$)
        ACCEPT 60,LENGHT
        TYPE 50
   50   FORMAT('   NUMBER OF ROWS FOR TRIM PICTURE-           ',$)
        ACCEPT 60,WIDTH
   60   FORMAT (I5)
C
        COLUM=(COLUM-1)*5
        WIDTH=WIDTH*5
        WIDTH=2*WIDTH
        ITEMP=PICX*5
        LENGHT=LENGHT*5
C
C       4 IS READ FILE, 3 IS WRITE FILE
C
        DEFINE FILE 4(0,ITEMP,U,POINT1)
        DEFINE FILE 3(WIDTH,LENGHT,U,POINT2)
        POINT2=1
        POINT1=ROW
        DO 70 I=1,WIDTH
        FIND (4'POINT1)
        READ(4'POINT1)(X(L),L=1,ITEMP)
        FIND(4'POINT1)
```

```
      LAST=COLUM+LENGHT-1
      WRITE(3'POINT2)(X(L),L=COLUM,LAST)
      FIND(3'POINT2)
70    CONTINUE
      STOP
      END
```

```
C
      PROGRAM TRIM2
C     ************************************************************
C     *                                                          *
C     *   TRIM2 TAKES OUT 120 ROWS AND 256 COLUMNS OF A          *
C     *   SPECIFIED 240 BY 512 PICTURE FILE AND WRITES           *
C     *   OUT THE DATA TO A SPECIFIED WRITE FILE.  THE           *
C     *   BEGINNING ROW AND COLUMN FOR THE RECTANGLE             *
C     *   WINDOW ARE INPUT BY THE USER.                          *
C     *                                                          *
C     ************************************************************
C     *                                                          *
C     *   SUBROUTINES REQUIRED:                                  *
C     *       FROM IOOP-RENTER, WENTRN, READR                    *
C     *       SEVERAL SYSLIB SUBROUTINES                         *
C     *                                                          *
C     ************************************************************
C
      LOGICAL*1 A(120,256),DB(12),AS(512)
      INTEGER*2 BUFF(256)
      EQUIVALENCE(AS(1),BUFF(1))
C
 10   ICH=IGETC()
      WRITE(5,20)
 20   FORMAT(1X,'INPUT THE READ FILE')
      CALL RENTER(ICH,DB)
      WRITE(5,30)
 30   FORMAT(1X,'INPUT THE WRITE FILE')
      ICW=IGETC()
      CALL WENTRN(ICW,60,DB)
      WRITE(5,40)
 40   FORMAT(1X,'INPUT THE UPPER LEFT HAND ROW AND
     1 COLUMN OF THE WINDOW',$)
      READ(5,50)IR,IC
 50   FORMAT(2I4)
C
C     READ IN THE WINDOWS, 60 ROWS AT A TIME AND WRITES
C     OUT 1 BLOCK AT A TIME.
C
      CALL READR(IR,IR+119,IC,IC+255,120,A,ICH,512)
      DO 70 K=1,60
      KK=(K-1)*2+1
      DO 60 J=1,256
      AS(J)=A(KK,J)
      AS(J+256)=A(KK+1,J)
 60   CONTINUE
      JS=IWRITW(256,BUFF,K-1,ICW)
 70   CONTINUE
      CALL CLOSEC(ICH)
      CALL CLOSEC(ICW)
      CALL IFREEC(ICH)
      CALL IFREEC(ICW)
      GO TO 10
      END
```

PDP-11 FORTRAN FILTER PROGRAMS AND MAJOR SUBROUTINES

```
      PROGRAM FILTER *
      INTEGER*2 XA(300)
      INTEGER WINDX,WINDY,PICX,PICY,IX,IY,ITEMP
      INTEGER POINT1,POINT2,X
      LOGICAL*1 A(600),MATRIX(20,20),B(600)
      COMMON WINDX,WINDY,PICX,PICY,IX,IY,POINT1,POINT2
      COMMON A ,MATRIX
              EQUIVALENCE (XA(1),B(1))
      CALL ASK
      CALL SIZE
      LX=PICX*.5
               LY=PICY*.5
       DEFINE FILE 4(0,LX,U,POINT1)
      DEFINE FILE 3(LY,PICX,U,POINT2)
      TYPE 10
10    FORMAT( '   WHAT ORDER FILTER    ',$)
      ACCEPT 11,K
11    FORMAT(3I)
      POINT2=1
      DO 40 I=1,PICY
      DO 20 J=1,PICX
      IX=J
      IY=I
      CALL WINDOW
      CALL FIND(K,X)
20    B(J)=X.AND.255
      I=I+1
      IP=I*.5
      DO 30 J=1,PICX
      IK=PICX+J
      IX=J
      IY=I
      CALL WINDOW
      CALL FIND(K,X)
30    B(IK)=X.AND.255
      WRITE(3'POINT2,ERR=40)(XA(J),J=1,PICX)
40    CONTINUE
      STOP
      END
```

*Main Program for MEDIAN and AVG Filters.

```
      SUBROUTINE FIND(K,X)*
     INTEGER B(400),I,J,L,R
      LOGICAL*1 A(600),MATRIX(20,20)
      INTEGER W,NUM,PICX,PICY,WINDX,WINDY,IX,IY,POINT1
      INTEGER POINT2,X
             COMMON WINDX,WINDY,PICX,PICY,IX,IY,POINT1
      COMMON POINT2,A,MATRIX
      DO 12 I=1,WINDY
      DO 21 J=1,WINDX
      B(J+(I-1)*WINDX)=255.AND.MATRIX(I,J)
21    CONTINUE
12    CONTINUE
      NUM=WINDX*WINDY
C     TYPE 234,(B(I),I=1,NUM)
       LSUM=0
      DO 355 I=1,NUM
355   LSUM=LSUM+B(I)
      X=LSUM/NUM
      RETURN
      END
```

*Called by AVG Filter.

```
      SUBROUTINE FIND(K,X)*
     INTEGER B(400),I,J,L,R
      LOGICAL*1 A(600),MATRIX(20,20)
      INTEGER W,NUM,PICX,PICY,WINDX,WINDY,IX,IY,POINT1
      INTEGER POINT2,X
            COMMON WINDX,WINDY,PICX,PICY,IX,IY,POINT1
      COMMON POINT2,A,MATRIX
      DO 12 I=1,WINDX
      DO 21 J=1,WINDY
      B(J+(I-1)*WINDY)=255.AND.MATRIX(I,J)
21    CONTINUE
12    CONTINUE
      NUM=WINDX*WINDY
C     TYPE 234,(B(I),I=1,NUM)
C234    FORMAT(1X,9(1X,I5))
      L=1
      R=NUM
2     IF(L.GT.R)GOTO 10
      X=B(K)
      I=L
      J=R
11    IF(B(I).GE.X)GOTO 19
      I=I+1
      GOTO 11
19    IF(X.GE.B(J))GOTO 20
      J=J-1
      GOTO 19
20    IF(I.GT.J) GOTO 30
      W=B(I)
      B(I)=B(J)
      B(J)=W
      I=I+1
      J=J-1
30    IF(I.LE.J) GOTO 11
      IF(J.LT.K) L=I
      IF(K.LT.I) R=J
      GOTO 2
C10    TYPE 234C,X
10    RETURN
      END
```

*Called by MEDIAN Filter.

```
      SUBROUTINE SIZE
      INTEGER WINDX,WINDY,PICX,PICY,IX,IY
      COMMON WINDX,WINDY,PICX,PICY,IX,IY
10    TYPE 100
      ACCEPT 200,PICX
      L=MOD(PICX,2)
      IF(L.NE.0) GOTO 10
15    TYPE 105
      ACCEPT 200,PICY
      L=MOD(PICY,2)
      IF (L.NE.0)GOTO 15
      TYPE 115
      ACCEPT 200,WINDX
      ITEMP=WINDX*.5
      WINDX=2*ITEMP+1
      TYPE 110
      ACCEPT 200,WINDY
      ITEMP=WINDY*.5
      WINDY=2*ITEMP+1
100   FORMAT('      #OF COLUMNS         ',$)
105   FORMAT('      #OF ROWS            ',$)
110   FORMAT('    ROWS OF WINDOW    ',$)
115   FORMAT('       COLS OF WINDOW       ',$)
200   FORMAT(I5)
      RETURN
      END
```

```
C       HUMAN VISUAL FILTER      AUGUST 8, 1977   MITCHELL
C       FILTERS A PICTURE FILE ASSUMING 256 POINTS PER ROW USING A
C       BANDPASS FILTER THAT SIMULATES THE HUMAN VISUAL SYSTEM
        DIMENSION       FILTER(45)
        INTEGER*2       JM(240),CENTER,ABOVE,BELOW,LEFT,RIGHT,SIZE
        LOGICAL*1       PIX(256,42),OUT(512),DUM(512)
        DATA    FILTER/0.,0.,0.,-.25,-.75,-.25,0.,0.,0.,0.,-.25,-.50,
     1.25,1.25,.25,-.5,-.25,0.,-.25,-.5,.25,1.,1.5,1.,.25,-.5,-.25,0.,
     1-.25,-.5,.25,1.25,.25,-.5,-.25,0.,0.,0.,-.25,-.75,-.25,0.,0.,
     10./
        IWINDO=4
C       JWINDO MUST BE EVEN BECAUSE THERE ARE 2 LINES PER DISK BLOCK
        JWINDO=2
C       JMOD MUST BE EVEN FOR 2 LINES PER DISK BLOCK CASE
        JMOD=JWINDO*2+2
        IF(JMOD.GT.42)GO TO 800
        DO 10 J=1,512
        DUM(J)=0
10      OUT(J)=0
C       SET UP MOD TABLE FOR Y POSITION
        DO 20 J=1,240
20      JM(J)=J-(J-1)/JMOD*JMOD
        CALL RENTER(ICR)
        CALL WENTRN(ICW,-1)
        IER=IREADW(256*JWINDO,PIX,0,ICR)
        IF(IER.LT.0)STOP 'BAD READ'
        DO 30 J=1,JWINDO/2
30      IER=IWRITW(256,DUM,J-1,ICW)
        NEXBLK=JWINDO-1
        DO 600 JODD=JWINDO+1,240-JWINDO,2
        NEXBLK=NEXBLK+1
        NEXLIN=JODD+JWINDO
        NEX=JM(NEXLIN)
C       READ IN ONE LINE INTO CIRCULAR BUFFER
        IER=IREADW(256,PIX(1,NEX),NEXBLK,ICR)
        IF(IER.LT.0)GO TO 650
        IADD=0
        DO 550 IHALF=1,2
        IF(IHALF.EQ.2)IADD=256
        J=JODD+IHALF-1
        DO 500 I=IWINDO+1,256-IWINDO
        INDEX=0
        SUM=0
        DO 50 J40=1,5
        K=JM(J-3+J40)
        DO 40 I40=1,9
        INDEX=INDEX+1
        IPOINT=PIX(I-5+I40,K).AND.255
40      SUM=SUM+IPOINT*FILTER(INDEX)
50      CONTINUE
        IOUT=SUM+64
        IF(IOUT.GT.255)IOUT=255
        IF(IOUT.LT.0)IOUT=0
500     OUT(I+IADD)=IOUT
550     CONTINUE
C
```

```
C
C
C       WRITE OUT LINE OF DATA
600     IER=IWRITW(256,OUT,J/2-1,ICW)
C
C
C       WRITE EMPTY LINES AT END
650     J=J/2-1
        DO 700 I=1,JWINDO/2
        J=J+1
700     IER=IWRITW(256,DUM,J,ICW)
        CALL CLOSEC(ICR)
        CALL CLOSEC(ICW)
        STOP 'ALL DONE'
800     WRITE(5,810)
810     FORMAT(1X,'VERTICAL WINDOW SIZE TOO LARGE')
        STOP
        END
```

```
C       MAX256          AUGUST 8, 1977          MITCHELL
C       FINDS LOCAL EXTREMA OVER A 256X240(OR LESS) PICTURE AND OUTPUTS THE
C       LOW ORDER 2 BITS OF EACH BYTE ARE CODED AS FOLLOWS
C       00 - HORZ MIN           01 - HORZ MAX
C       10 - VERT MIN           11 - VERT MAX
C       EXTREMA SIZE INDICATED BY HIGH 6 BITS
C       MAXMIN SEARCHES OVER A WINDOW OF IWINDO X JWINDO AROUND
C       EACH POINT INDICATING WHETHER IT IS A LOCAL EXTREME AND
C       ITS SIZE.  THE LOW ORDER BIT IS A ONE IF EXTREME IS A MAXIMUM
C       AND THE NEXT BIT IS A ONE IF IT IS A VERTICAL EXTREME.
        INTEGER*2       JM(240),CENTER,ABOVE,BELOW,LEFT,RIGHT,SIZE
        LOGICAL*1       PIX(256,42),OUT(512) DUM(512)
        IWINDO=24
C       JWINDO MUST BE EVEN FOR 2 LINES FOR DISK BLOCK CASE
        JWINDO=12
C       JMOD MUST BE EVEN FOR 2 LINES PER DISK BLOCK CASE
        JMOD=JWINDO*2+2
        IF(JMOD.GT.42)GO TO 800
        DO 10 J=1,512
        DUM(J)=0
10      OUT(J)=0
C       SET UP MOD TABLE FOR Y POSITION
        DO 20 J=1,240
20      JM(J)=J-(J-1)/JMOD*JMOD
        CALL RENTER(ICR)
        CALL WENTRN(ICW,-1)
        IER=IREADW(256*JWINDO,PIX,0,ICR)
        IF(IER.LT.0)STOP 'BAD READ'
        DO 30 J=1,JWINDO/2
30      IER=IWRITW(256,DUM,J-1,ICW)
        NEXBLK=JWINDO-1
        DO 600 JODD=JWINDO+1,240-JWINDO,2
        NEXBLK=NEXBLK+1
        NEXLIN=JODD+JWINDO
        NEX=JM(NEXLIN)
C       READ IN ONE LINE INTO CIRCULAR BUFFER
        IER=IREADW(256,PIX(1,NEX),NEXBLK,ICR)
        IF(IER.LT.0)GO TO 650
        IADD=0
        DO 550 IHALF=1,2
        IF(IHALF.EQ.2)IADD=256
        J=JODD+IHALF-1
        DO 500 I=IWINDO+1,256-IWINDO
        K1=JM(J)
        CENTER=PIX(I,K1).AND.255
        K=JM(J-1)
        ABOVE=PIX(I,K).AND.255
        K=JM(J+1)
        BELOW=PIX(I,K).AND.255
        SIZE=0
        IF((CENTER.GT.ABOVE).AND.(CENTER.GE.BELOW))GO TO 100
        IF((CENTER.LT.ABOVE).AND.(CENTER.LE.BELOW))GO TO 200
50      LEFT=PIX(I-1,K1).AND.255
        RIGHT=PIX(I+1,K1).AND.255
        IF((CENTER.GT.LEFT).AND.(CENTER.GE.RIGHT))GO TO 300
```

```
      IF((CENTER.LT.LEFT).AND.(CENTER.LE.RIGHT))GO TO 400
      GO TO 500
C
C
C     VERTICAL MAXIMUM DETERMINATION
100   DO 130 K=J+2,J+JWINDO
      L=JM(K)
      NEXT=PIX(I,L).AND.255
      IF(NEXT.GE.CENTER)GO TO 105
      IF(NEXT.LT.BELOW)GO TO 120
      GO TO 130
105   IF(NEXT.EQ.BELOW)GO TO 130
C     FINISHED
110   K=J+JWINDO
      GO TO 130
C     NEW LOW
120   BELOW=NEXT
130   CONTINUE
      DO 160 K=J-JWINDO,J-2
      M=J-JWINDO+J-2-K
      L=JM(M)
      NEXT=PIX(I,L).AND.255
      IF(NEXT.GE.CENTER)GO TO 140
      IF(NEXT.LT.ABOVE)GO TO 150
      GO TO 160
C     FINISHED
140   K=J-2
      GO TO 160
C     NEW LOW
150   ABOVE=NEXT
160   CONTINUE
      IF(ABOVE.GT.BELOW)BELOW=ABOVE
      SIZE=CENTER-BELOW
      SIZE=SIZE*2
      IF(SIZE.GT.255)SIZE=255
      SIZE=SIZE.AND.252
      SIZE=SIZE+3
C     INDICATES VERTICAL MAXIMUM
      GO TO 50
C
C
C     VERTICAL MINIMUM DETERMINATION
200   DO 230 K=J+2,J+JWINDO
      L=JM(K)
      NEXT=PIX(I,L).AND.255
      IF(NEXT.LE.CENTER)GO TO 205
      IF(NEXT.GT.BELOW)GO TO 220
      GO TO 230
205   IF(NEXT.EQ.BELOW)GO TO 230
C     FINISHED
210   K=J+JWINDO
      GO TO 230
C     NEW HIGH
220   BELOW=NEXT
230   CONTINUE
```

```
      DO 260 K=J-JWINDO,J-2
      M=J-JWINDO+J-2-K
      L=JM(M)
      NEXT=PIX(I,L).AND.255
      IF(NEXT.LE.CENTER)GO TO 240
      IF(NEXT.GT.ABOVE)GO TO 250
      GO TO 260
C     FINISHED
240   K=J-2
      GO TO 260
C     NEW LOW
250   ABOVE=NEXT
260   CONTINUE
      IF(ABOVE.GT.BELOW)ABOVE=BELOW
      SIZE=ABOVE-CENTER
      SIZE=SIZE*2
      IF(SIZE.GT.255)SIZE=255
      SIZE=SIZE.AND.252
      SIZE=SIZE+2
C     INDICATES VERTICAL MINIMUM
      GO TO 50
C
C
C     HORIZONTAL MAXIMUM DETERMINATION
300   DO 330 L=I+2,I+IWINDO
      NEXT=PIX(L,K1).AND.255
      IF(NEXT.GE.CENTER)GO TO 305
      IF(NEXT.LT.RIGHT)GO TO 320
      GO TO 330
305   IF(NEXT.EQ.RIGHT)GO TO 330
C     FINISHED
310   L=I+IWINDO
      GO TO 330
C     NEW LOW
320   RIGHT=NEXT
330   CONTINUE
      DO 360 L=I-IWINDO,I-2
      M=I-IWINDO+I-2-L
      NEXT=PIX(M,K1).AND.255
      IF(NEXT.GE.CENTER)GO TO 340
      IF(NEXT.LT.LEFT)GO TO 350
      GO TO 360
C     FINISHED
340   L=I-2
      GO TO 360
C     NEW LOW
350   LEFT=NEXT
360   CONTINUE
      IF(LEFT.GT.RIGHT)RIGHT=LEFT
      ISIZE=CENTER-RIGHT
      ISIZE=ISIZE*2
      IF(ISIZE.GT.255)ISIZE=255
      IF(SIZE.GT.ISIZE)GO TO 500
      SIZE=ISIZE.AND.252
      SIZE=SIZE+1
C     INDICATES HORIZONTAL MAXIMUM
```

```
        GO TO 500
C
C
C       HORIZONTAL MINIMUM DETERMINATION
400     DO 430 L=I+2,I+IWINDO
        NEXT=PIX(L,K1) AND 255
        IF(NEXT.LE.CENTER)GO TO 405
        IF(NEXT.GT.RIGHT)GO TO 420
        GO TO 430
405     IF(NEXT.EQ.RIGHT)GO TO 430
C       FINISHED
410     L=I+IWINDO
        GO TO 430
C       NEW HIGH
420     RIGHT=NEXT
430     CONTINUE
        DO 460 L=I-IWINDO,I-2
        M=I-IWINDO+I-2-L
        NEXT=PIX(M,K1).AND.255
        IF(NEXT.LE.CENTER)GO TO 440
        IF(NEXT.GT.LEFT)GO TO 450
        GO TO 460
C       FINISHED
440     L=I-2
        GO TO 460
C       NEW LOW
450     LEFT=NEXT
460     CONTINUE
        IF(LEFT.GT.RIGHT)LEFT=RIGHT
        ISIZE=LEFT-CENTER
        ISIZE=ISIZE**2
        IF(ISIZE.GT.255)ISIZE=255
        IF(SIZE.GT.ISIZE)GO TO 500
        SIZE=ISIZE.AND.252
C       INDICATES HORIZONTAL MINIMUM
C
C
C
C
C
C       WRITE OUT POINT
500     OUT(I+IADD)=SIZE
550     CONTINUE
C
C
C
C       WRITE OUT LINE OF DATA
600     IER=IWRITW(256,OUT,J/2-1,ICW)
C
C
C       WRITE EMPTY LINES AT END
650     J=J/2-1
        DO 700 I=1,JWINDO/2
        J=J+1
700     IER=IWRITW(256,DUM,J,ICW)
```

```
      CALL CLOSEC(ICR)
      CALL CLOSEC(ICW)
      STOP 'ALL DONE'
800   WRITE(5,810)
810   FORMAT(1X,'VERTICAL WINDOW SIZE TOO LARGE')
      STOP
      END
```

```
C       MAX512          AUGUST 8, 1977          MITCHELL
C       FINDS LOCAL EXTREMA OVER A 512X240 PICTURE AND OUTPUTS THEM
C       LOW ORDER 2 BITS OF EACH BYTE ARE CODED AS FOLLOWS
C       00 - HORZ MIN           01 - HORZ MAX
C       10 - VERT MIN           11 - VERT MAX
C       EXTREMA SIZE INDICATED BY HIGH 6 BITS
C       MAXMIN SEARCHES OVER A WINDOW OF IWINDO X JWINDO AROUND
C       EACH POINT INDICATING WHETHER IT IS A LOCAL EXTREME AND
C       ITS SIZE.  THE LOW ORDER BIT IS A ONE IF EXTREME IS A MAXIMUM
C       AND THE NEXT BIT IS A ONE IF IT IS A VERTICAL EXTREME.
        INTEGER*2       JM(240),CENTER,ABOVE,BELOW,LEFT,RIGHT,SIZE
        LOGICAL*1       PIX(512,41),OUT(512),DUM(512)
        IWINDO=30
        JWINDO=15
        JMOD=JWINDO*2+1
        IF(JMOD.GT.41)GO TO 800
        DO 10 J=1,512
        DUM(J)=0
10      OUT(J)=0
C       SET UP MOD TABLE FOR Y POSITION
        DO 20 J=1,240
20      JM(J)=J-(J-1)/JMOD*JMOD
        CALL RENTER(ICR)
        CALL WENTRN(ICW,-1)
        IER=IREADW(512*JWINDO,PIX,0,ICR)
        IF(IER.LT.0)STOP 'BAD READ'
        DO 30 J=1,JWINDO
30      IER=IWRITW(256,DUM,J-1,ICW)
        DO 600 J=JWINDO+1,240-JWINDO
        NEXLIN=J+JWINDO
        NEX=JM(NEXLIN)
C       READ IN ONE LINE INTO CIRCULAR BUFFER
        IER=IREADW(256,PIX(1,NEX),NEXLIN-1,ICR)
        IF(IER.LT.0)GO TO 650
        DO 500 I=IWINDO+1,512-IWINDO
        K1=JM(J)
        CENTER=PIX(I,K1).AND.255
        K=JM(J-1)
        ABOVE=PIX(I,K).AND.255
        K=JM(J+1)
        BELOW=PIX(I,K).AND.255
        SIZE=0
        IF((CENTER.GT.ABOVE).AND.(CENTER.GE.BELOW))GO TO 100
        IF((CENTER.LT.ABOVE).AND.(CENTER.LE.BELOW))GO TO 200
50      LEFT=PIX(I-1,K1).AND.255
        RIGHT=PIX(I+1,K1).AND.255
        IF((CENTER.GT.LEFT).AND.(CENTER.GE.RIGHT))GO TO 300
        IF((CENTER.LT.LEFT).AND.(CENTER.LE.RIGHT))GO TO 400
        GO TO 500
C
C
C       VERTICAL MAXIMUM DETERMINATION
100     DO 130 K=J+2,J+JWINDO
        L=JM(K)
        NEXT=PIX(I,L).AND.255
```

```
      IF(NEXT.GE.CENTER)GO TO 105
      IF(NEXT.LT.BELOW)GO TO 120
      GO TO 130
C     CHECK FOR FLAT TOP
105   IF(NEXT.EQ.BELOW)GO TO 130
C     FINISHED
110   K=J+JWINDO
      GO TO 130
C     NEW LOW
120   BELOW=NEXT
130   CONTINUE
      DO 160 K=J-JWINDO,J-2
      M=J-JWINDO+J-2-K
      L=JM(M)
      NEXT=PIX(I,L).AND.255
      IF(NEXT.GE.CENTER)GO TO 140
      IF(NEXT.LT.ABOVE)GO TO 150
      GO TO 160
C     FINISHED
140   K=J-2
      GO TO 160
C     NEW LOW
150   ABOVE=NEXT
160   CONTINUE
      IF(ABOVE.GT.BELOW)BELOW=ABOVE
      SIZE=CENTER-BELOW
      SIZE=SIZE*2
      IF(SIZE.GT.255)SIZE=255
      SIZE=ISIZE.AND.252
      SIZE=SIZE+3
C     INDICATES VERTICAL MAXIMUM
      GO TO 50
C
C
C     VERTICAL MINIMUM DETERMINATION
200   DO 230 K=J+2,J+JWINDO
      L=JM(K)
      NEXT=PIX(I,L).AND.255
      IF(NEXT.LE.CENTER)GO TO 205
      IF(NEXT.GT.BELOW)GO TO 220
      GO TO 230
205   IF(NEXT.EQ.BELOW)GO TO 230
C     FINISHED
210   K=J+JWINDO
      GO TO 230
C     NEW HIGH
220   BELOW=NEXT
230   CONTINUE
      DO 260 K=J-JWINDO,J-2
      M=J-JWINDO+J-2-K
      L=JM(M)
      NEXT=PIX(I,L).AND.255
      IF(NEXT.LE.CENTER)GO TO 240
      IF(NEXT.GT.ABOVE)GO TO 250
      GO TO 260
```

```
C       FINISHED
240     K=J-2
        GO TO 260
C       NEW LOW
250     ABOVE=NEXT
260     CONTINUE
        IF(ABOVE GT BELOW)ABOVE=BELOW
        SIZE=ABOVE-CENTER
        SIZE=SIZE*2
        IF(SIZE GT 255)SIZE=255
        SIZE=SIZE AND 252
        SIZE=SIZE+2
C       INDICATES VERTICAL MINIMUM
        GO TO 50
C
C
C       HORIZONTAL MAXIMUM DETERMINATION
300     DO 330 L=I+2, I+IWINDO
        NEXT=PIX(L,K1) AND 255
        IF(NEXT GE CENTER)GO TO 305
        IF(NEXT LT RIGHT)GO TO 320
        GO TO 330
305     IF(NEXT EQ RIGHT)GO TO 330
C       FINISHED
310     L=I+IWINDO
        GO TO 330
C       NEW LOW
320     RIGHT=NEXT
330     CONTINUE
        DO 360 L=I-IWINDO, I-2
        M=I-IWINDO+I-2-L
        NEXT=PIX(M,K1) AND 255
        IF(NEXT GE CENTER)GO TO 340
        IF(NEXT LT LEFT)GO TO 350
        GO TO 360
C       FINISHED
340     L=I-2
        GO TO 360
C       NEW LOW
350     LEFT=NEXT
360     CONTINUE
        IF(LEFT GT RIGHT)RIGHT=LEFT
        ISIZE=CENTER-RIGHT
        ISIZE=ISIZE*2
        IF(ISIZE GT 255)ISIZE=255
        IF(SIZE GT ISIZE)GO TO 500
        SIZE=SIZE AND 252
        SIZE=SIZE+1
C       INDICATES HORIZONTAL MAXIMUM
        GO TO 500
C
C
C       HORIZONTAL MINIMUM DETERMINATION
400     DO 430 L=I+2, I+IWINDO
        NEXT=PIX(L,K1) AND 255
        IF(NEXT LE CENTER)GO TO 405
```

```
      IF(NEXT.GT.RIGHT)GO TO 420
      GO TO 430
405   IF(NEXT.EQ.RIGHT)GO TO 430
C     FINISHED
410   L=I+IWINDO
      GO TO 430
C     NEW HIGH
420   RIGHT=NEXT
430   CONTINUE
      DO 460 L=I-IWINDO,I-2
      M=I-IWINDO+I-2-L
      NEXT=PIX(M,K1).AND.255
      IF(NEXT.LE.CENTER)GO TO 440
      IF(NEXT.GT.LEFT)GO TO 450
      GO TO 460
C     FINISHED
440   L=I-2
      GO TO 460
C     NEW LOW
450   LEFT=NEXT
460   CONTINUE
      IF(LEFT.GT.RIGHT)LEFT=RIGHT
      ISIZE=LEFT-CENTER
      ISIZE=ISIZE*2
      IF(ISIZE.GT.255)ISIZE=255
      IF(SIZE.GT.ISIZE)GO TO 500
      SIZE=ISIZE.AND.252
C     INDICATES HORIZONTAL MINIMUM
C
C
C
C
C
C     WRITE OUT POINT
500   OUT(I)=SIZE
C
C
C
C     WRITE OUT LINE OF DATA
600   IER=IWRITW(256,OUT,J-1,ICW)
C
C
C     WRITE EMPTY LINES AT END

640   GO TO 660
650   J=J-1
660   DO 700 I=1,JWINDO
      J=J+1
700   IER=IWRITW(256,DUM,J-1,ICW)
      CALL CLOSEC(ICR)
      CALL CLOSEC(ICW)
      STOP 'ALL DONE'
800   WRITE(5,810)
810   FORMAT(1X,'VERTICAL WINDOW SIZE TOO LARGE')
      STOP
      END
```

PDP-11 FORTRAN INPUT/OUTPUT SUBROUTINES

```
      SUBROUTINE WRITEF(LU,ICH,DB)
C
C     ************************************************************
C     *                                                          *
C     *   WRITEF CONNECTS THE USER ENTERED INPUT FILE TO         *
C     *   CHANNEL ICH AND THE OUTPUT FILE TO LOGICAL UNIT        *
C     *   LU.  LU IS ONLY INPUT PARAMETER WHILE ICH AND          *
C     *   THE 12 BYTE ARRAY DB ARE RETURNED ON OUTPUT.           *
C     *   ARRAY DB WILL CONTAIN THE ASCII EQUIVALENT OF          *
C     *   THE INPUT FILE NAME.                                   *
C     *                                                          *
C     ************************************************************
C
      LOGICAL*1 DB(12)
      INTEGER*2 SPEC(39),DEF(4)
      COMMON/ICS/SPEC,DEF
      DATA DEF/3RDAT,3RDAT,3RLST,3RLST/
C
C     CALL RT-11 CSI TO INTERPRET THE INPUT COMMAND STRING
C
 10   WRITE(5,1)
 1    FORMAT(' ENTER OUTPUT AND INPUT FILES (CSI FORMAT)'/)
      IF(ICSI(SPEC,DEF,,,0).NE.0) GO TO 10
C
C     ALLOCATE AN RT-11 CHANNEL BETWEEN 0 AND 17
C           ICH=-1 - NO CHANNELS AVAILABLE
C              = N - CHANNEL N HAS BEEN ALLOCATED
C
      ICH=IGETC()
      IF(ICH.EQ.-1)STOP ' NO CHANNELS AVAILABLE'
C
C     CONVERT THE 12 CHARACTER RADIX-50 EQUIVALENT OF THE
C     INPUT FILE NAME IN SPEC(16) THROUGH SPEC(17) TO THE
C     ASCII EQUIVALENT IN DB(1) THROUGH DB(12)
C
      JS=R50ASC (12,SPEC(16),DB)
C
C     ASSOCIATE ALLOCATED CHANNEL WITH INPUT FILE
C         JS = -1 - CHANNEL ALREADY OPEN (SHOULDN'T OCCUR)
C            = -2 - FILE SPECIFIED NOT ON DEVICE
C            =  N = NUMBER OF BLOCKS IN INPUT FILE
C                   (NORMAL RETURN)
      JS=LOOKUP(ICH,SPEC(16))
      IF(JS.GE.0) GO TO 20
      IF(JS.EQ.-1) STOP ' CHANNEL ALREADY OPEN'
      IF(JS.EQ.-2) WRITE(5,3)
 3    FORMAT(' FILE SPECIFIED NOT ON DEVICE'/)
      GO TO 10
 20   WRITE(5,5) DB,JS
 5    FORMAT(' INPUT FILE ',9A1,'.',3A1,' HAS ',I3,' BLOCKS'/)
C
C     BEFORE OUTPUT FILE IS OPENED, PLACE THE NECESSARY I/O
C     FILE INFORMATION IN THE FORTRAN LOGICAL UNIT TABLE
C           JS = 0  - NORMAL RETURN
C              ELSE - UNIT ALREADY OPEN OR THERE IS NO SPACE
C                     FOR ANOTHER LOGICAL UNIT ASSOCIATION
```

```
      JJ=IASIGN(LU,SPEC(1),SPEC(2),SPEC(5),3)
      IF(JJ.EQ.0) RETURN
      WRITE(5,6) LU
    6 FORMAT(/,' LOGICAL UNIT ',I2,' IS EITHER ALREADY OPEN',
     1/,' OR THERE IS NO ROOM FOR ANOTHER LOGICAL UNIT',/,
     2' ASSOCIATION ')
      STOP
      END
```

```
      SUBROUTINE READR(IRB,IRF,ICB,ICF,NR,A,ICH,NC)
C READR READS A SPECIFIED WINDOW OF ROWS AND
C COLUMNS FROM A FILE CONNECTED TO CHANNEL NO
C ICH INTO THE PARAMETER ARRAY A. THE INPUT FILE
C SHOULD CONTAIN SEQUENTIAL ROWS OF NC BYTES EACH
C A SHOULD BE DIMENSIONED WITH NR ROWS IN
C THE CALLING PROGRAM
C
C PARAMETERS
C       IRB = THE BEGINNING ROW
C       IRF = THE ENDING ROW
C       ICB = THE BEGINNING COLUMN
C       ICF = THE ENDING COLUMN
C       NR  = THE NUMBER OF ROWS IN THE MAIN
C             PROGRAM DIMENSION STATEMENT FOR A
C       A   = THE LOGICAL*1 2-DIM DATA ARRA
C       ICH = THE CHANNEL CONNECTED TO THE INPUT
C             FILE
C
      INTEGER*2 BUFF(256)
      LOGICAL*1 A(1),AS(512)
      EQUIVALENCE(AS(1),BUFF(1))
      CC=512/NC
      IRR=IRB-1
C FIND BLOCK WHICH HOLDS FIRST ROW
      B=IRR/CC
      IB=B
      K=0
      IF((B-IB).GT.0.0.AND.NC.GT.255)K=256
C COMPUTE NUMBER OF COLUMNS AND ROWS TO BE READ
      IR=IRF-IRR
      IC=ICF-ICB+2
C READ IN FIRST BLOCK OF DATA
      JS=IREADW(256,BUFF,IB,ICH)
C BEGIN TO TRANSFER THE PROPER COLUMNS OF THE
C CURRENT ROW TO THE ARRAY
      DO 30 JR=1,IR
      DO 20 JC=1,IC
      A((JC-1)*NR+JR)=AS(K+ICB)
      K=K+1
20    CONTINUE
C CHECK FOR NEW BLOCK
      IF(NC.GE.256)GO TO 25
      IF(K.GE.512)GO TO 40
30    CONTINUE
      RETURN
40      IB=IB+1
      JS=IREADW(256,BUFF,IB,ICH)
      K=0
      GO TO 30
25    IF(K.GE.256.OR.CC.EQ.1.0)GO TO 40
      K=256
      GO TO 30
      END
```

```
      SUBROUTINE RENTER(ICH,PB)
C RENTER INTERROGATES THE USER FOR
C THE INPUT FILE IN STANDARD CSI FILE FORMAT
C AND CONNECTS THE FILE TO A FREE CHANNEL
C NUMBER THE CHANNEL AND 12 BYTE ASCII FILE
C NAME ARE PASSED BACK THROUGH ICH AND DB.
C
C PARAMETERS
C       ICH = THE CHANNEL TO WHICH THE FILE
C             WAS CONNECTED
C       DB  = THE 12 BYTE VECTOR CONTAINING
C             THE FILE ASCII NAME
C
      LOGICAL*1 DB(12),IDU,PB(12)
      INTEGER*2 DBB(6),IB(4),SPEC(1),DEF(1)
      EQUIVALENCE (DBB(1),DB(1)),(SPEC(16),IB(1))
      COMMON /ICS/SPEC,DEF
10    WRITE(5,3)
3     FORMAT(1X,'ENTER A CSI INPUT FILE DESCRIPTOR ')
      IF(ICSI(SPEC,DEF,,,0).NE.0)GO TO 10
      ICH=IGETC()
      IF(ICH.LT.0)STOP 'NO CHANNELS AVAILABLE BUD'
      JS=R50ASC(12,IB,DBB)
      JS=LOOKUP(ICH,IB)
      IF(JS.LT.0)WRITE(5,5)
      IF(JS.LT.0)GO TO 10
5     FORMAT(1X,'NO SUCH INPUT FILE BUD')
      WRITE(5,6)JS
6     FORMAT(1X,'INPUT FILE ',I3,' BLOCKS')
      DO 20 II=1,12
      PB(II)=DB(II)
20    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE WENTRN(ICH,NB,PB)
C WENTRN INTERROGATES THE USER AT THE TTY
C FOR THE STANDARD CSI OUTPUT FILE NAME.
C IT THEN CONNECTS UP THE FILE TO A FREE CHANNEL
C NUMBER AND MAKES A TENTATIVE ENTRY ON
C THE DIRESTORY OF THE APPROPRIATE DEVICE.
C THE FILE IS MADE PERMANENT BY THE USER BY
C CLOSING THE CHANNEL.
C
C PARAMETERS :
C       ICH = THE CHANNEL TO SHICH THE FILE WAS
C             CONNECTED
C       NB = THE NUMBER OF BLOCKS TO RESERVE FOR THE
C            OUTPUT FILE
C       DB = THE 12 BYTE VECTOR CONTAINING THE FILE
C            ASCII NAME
C
      LOGICAL*1 DB(12),IDU,PB(12)
      INTEGER*2 DBB(6),IB(4),SPEC(1),DEF(1)
      EQUIVALENCE (DB(1),DBB(1)),(SPEC(16),IB(1))
      COMMON /ICS/SPEC,DEF
10    WRITE(5,3)
3     FORMAT(1X,'ENTER A CSI OUTPUT FILE DESCRIPTOR'/)
      IF(ICSI(SPEC,DEF,,,0).NE.0)GO TO 10
      ICH=IGETC()
      IF(ICH.LT.0)STOP 'NO CHANNELS AVAILABLE BUD'
      JS=R50ASC(12,IB,DBB)
      JS=IENTER(ICH,IB,NB)
      IF(JS.LT.0)WRITE(5,5)
5     FORMAT(1X,'NO SPACE ON THE DEV BUD')
      IF(JS.LT.0)GO TO 10
      WRITE(5,6)JS
6     FORMAT(1X,'OUTPUT FILE RESERVED ',I3,' BLOCKS')
      DO 20 II=1,12
      PB(II)=DB(II)
20    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE AREAD(IRB, IRF, ICB, ICF, NR, A, ICH)
      INTEGER*2 BUFF(256)
      LOGICAL*1 AS(512), A(1)
      EQUIVALENCE (AS(1), BUFF(1))
C FIND FIRST BLOCK TO BE READ USING 240 BYTE COL
      ICC=ICB-1
      IB=ICC/2 125
C COMPUTE NUMBER OF COLUMNS AND ROWS TO BE READ
      IR=IRF-IRB+1
      IC=ICF-ICC
C READ IN A BLOCK OF DATA
      JS=IREADW(256, BUFF, IB, ICH)
C READ IN COLUMNS TO A
      DO 30 JC=1, IC
C COMPUTE POSITION IN THE BLOCK OF THE FIRST
C BYTE TO BE READ
      P=FLOAT(JC+ICC-1)*240. -(FLOAT(IB)*512 0)
      IP=P+IRB-1 0
      I=1
C READ IN THE ROW ELEMENTS FROM THE BUFFER IF THE END
C OF THE BUFFER IS REACHED JUMP TO 100
      DO 20 JR=1, IR
      IF(IP+I GT 512)GO TO 100
10    A((JC-1)*NR+JR)=AS(IP+I)
      I=I+1
20    CONTINUE
30    CONTINUE
      RETURN
C INCREMENT BLOCK COUNTER, READ IN A NEW BLOCK AND
C SET CURRENT BYTE INDEX TO ZERO
100   IB=IB+1
      JS=IREADW(256, BUFF, IB, ICH)
      IP=IP-512
      IF(IP LT 0)IP=0
      I=1
      GO TO 10
      END
```

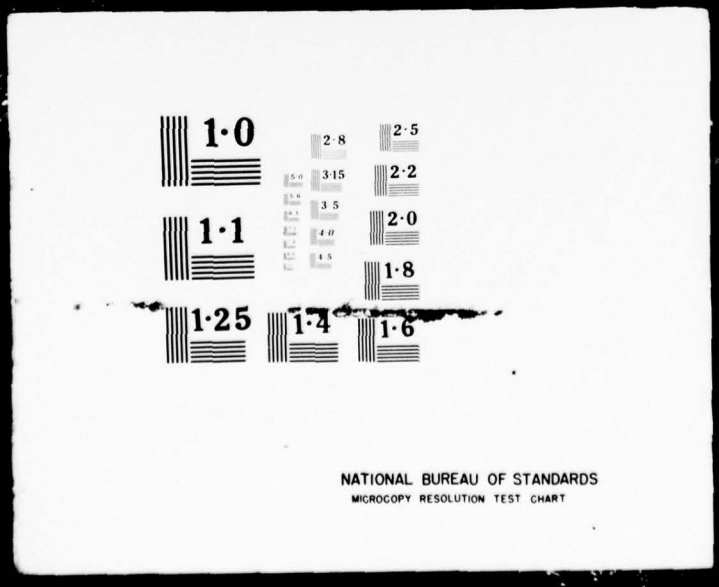
```
      SUBROUTINE ASK
      INTEGER SPEC(39)
      REAL*4 EXT(2)
      DATA EXT/6RDATIDAT,6RDATDAT/
10    TYPE 99
99    FORMAT('  ENTER OUTPUT AND INPUT FILES  ')
C
      IF(ICSI(SPEC,EXT,,,0).NE.0)GOTO 10
C     3 IS WRITE FILE
      CALL IASIGN(3,SPEC(1),SPEC(2),SPEC(5),1)
C
C        4 IS READ FILE
C
      CALL IASIGN(4,SPEC(16),SPEC(17),0,32)
      RETURN
      END
      SUBROUTINE WINDOW
      INTEGER HAVE
     LOGICAL*1 A(600),MATRIX(20,20),B(20,600)
      INTEGER X(600)
      INTEGER WINDX,WINDY,PICX,PICY,IX,IY,ITEMP
      INTEGER POINT1,POINT2
      COMMON WINDX,  WINDY,PICX,PICY,IX,IY,POINT1
      COMMON POINT2,A,MATRIX
      EQUIVALENCE(A(19),X(1))
C
      IF(IY.EQ.HAVE) GOTO 97
      J=0
C
C     J IS THE NUMBER OF DUMMY ROWS
C
      JKTEMP=.5*WINDY
      K=IY-JKTEMP
      IF(K) 100,100,200
 100  DO 35 I=K,0
  35  J=J+1
 200  POINT1=K+J
      FIND(4'POINT1)
C
C     FIRST ROW TO BE READ IS POINT1
C
      ITEMP=.5*WINDX
      JK=J+1
      LWINDX=.5*PICX
      DO 17 K=JK,WINDY
      READ(4'POINT1,END=37) (X(I),I=1,LWINDX)
C     TYPE 47,(X(I),I=1,40)
      DO 27 L=1,WINDX
27    MATRIX(K,L)=A(IX-ITEMP+L+17)
      DO 87 I=19,600
87    B(K,I)=A(I)
17    CONTINUE
C     TYPE 47,((MATRIX(I,J),J=1,WINDX),I=1,WINDY)
37    CONTINUE
      HAVE=IY
```

AD-A048 156 WHITE SANDS MISSILE RANGE N MEX INSTRUMENTATION DIRE--ETC F/G 17/8
SEGMENTATION AND STRUCTURE ANALYSIS FOR REAL-TIME VIDEO TARGET --ETC(U)
OCT 77 K FUKUNAGA , A L GILBERT, M K GILES

UNCLASSIFIED STEWS-ID-77-1 NL

3 OF 3
AD
A048156

END
DATE
FILMED
1-78
DDC



```
        RETURN
97      DO 107 J=JK,WINDY
        DO 117 I=1,WINDX
117     MATRIX(J,I)=B(J,IX-ITEMP+I+17)
107     CONTINUE
C       TYPE 47,((MATRIX(I,J),J=1,WINDX),I=1,WINDY)
C       TYPE 47,(B(I),I=1,50)
47      FORMAT(1X,8(1X,08))
        RETURN
        END
```

PDP-11 FORTRAN GRAPHICS SUBROUTINES

```
      SUBROUTINE START(LL)
C START INITIALIZES ALL THE COMMON VARIABLES
C USED BY THE GRAPH LIBRARY ROUTINES.
C ALL CALLS TO GRAPH ROUTINES WHICH
C CONTAIN LABELED COMMON BLOCK GRAP
C SHOULD BE PRECEDED BY A CALL TO START.
C
C PARAMETERS :
C       LL = THE LOGICAL UNIT NUMBER TO WHICH
C            ALL GRAPHICS OUTPUT WILL BE
C            SENT
C
C COMMON VARIABLES :
C       LU = THE LOGICAL UNIT NUMBER TO WHICH
C            THE OUTPUT WILL BE SENT
C       IX = THE X COORDINATE OF THE LOWER LEFT
C             HAND CORNER OF THE GRAPHICS WINDOW
C       IY = THE Y COORDINATE OF THE LOWER LEFT
C            HAND CORNER OF THE GRAPHICS WINDOW
C       ISX = THE SIZE IN THE X DIRECTION
C             OF THE WINDOW
C       ISY = THE SIZE IN THE Y DIRECTION OF THE
C             WINDOW
C       A   = THE LOGICAL*1 ARRAY USED TO OUTPUT
C            THE COMMANDS TO DEVICE LU
C
      LOGICAL*1 A(2000)
      COMMON /GRAP/LU, IX, IY, ISX, ISY, A
      DATA A/2000*0/
      LU=LL
      IX=100
      IY=100
      ISX=512
      ISY=512
      RETURN
      END
```

```
      SUBROUTINE GRA(IR,IC,NR,A)
C GRA PLOTS OUT THE ROWS AND COLUMNS
C CONTAINED IN THE BYTE ARRAY A.
C THE VALUES IN A ARE FIRST SCALED TO
C 16 "GREY" LEVELS BETWEEN :
C          IB < GL < IB+IS
C WHERE IB AND IS ARE ENTERED AT THE TTY
C GSX THEN PLOTS OUT THE CORRESPONDING
C GREY VALUES BY ROWS. THE LEFT BOTTOM
C CORNER OF THE WINDOW IS AT
C   X = 620
C   Y = 360
C
C PARAMETERS :
C        IR = THE NUMBER OF ROWS IN A
C        IC = THE NUMBER OF COLUMNS IN A
C        NR = THE NUMBER OF ROWS IN THE DIMENSIONING
C             OF A
C        A  = THE DATA ARRAY LOGICAL*1
C
C SUBROUTINES REQUIRED :
C        FROM GRAPH - GSX,LOADB,LX,CKHX
C
C LIMITS :
C        A CAN HAVE AT MOST 100 ROWS
C
      LOGICAL*1 A(NR,1),IZ(100),K,B(1)
      COMMON /GRAP/LU,IX,IY,ISX,ISY,B
      WRITE(5,1)
1     FORMAT(1X,'INPUT YOUR VALUE FOR BLACK AND
     1 THE CONTRAST SPREAD PLEASE 2I4')
      READ(5,2)IB,IS
2     FORMAT(2I4)
      CALL INIT(5)
      S=IS
      DO 20 J=1,IR
      II=IR-J+1
      DO 10 I=1,IC
      JJ=IC-I+1
      KK=A(II,JJ).AND.255
      FK=KK-IB
      IF(FK.LT.0)FK=0
      IF(FK.GT.S)FK=S
      IZ(I)=16.-(16.0*FK/S)
10    CONTINUE
      CALL GSX(155,90+J,IZ,IC)
20    CONTINUE
      CALL LOADB(620,300,B,2)
      IS=IB+IS-1
      B(1)=29
      B(6)=31
      WRITE(LU,3)(B(I),I=1,6),IB,IS
3     FORMAT('+',$,6A1,'THE CONTRAST SPREAD
     1  BLACK = ',I3,2X,'WHITE = ',I3)
      CALL LOADB(0,760,B,2)
      WRITE(LU,4)(B(I),I=1,6)
```

```
4       FORMAT('+',$,6A1)
        RETURN
        END
```

```
      SUBROUTINE GSX(IX,IY,IZ,NZ)
C GSX FORMS A GREY LEVEL INTENSITY PLOT
C FOR A ROW OF VALUES PASSED THROUGH THE
C BYTE VECTOR IZ. THE VALUES IN IV MUST
C BE BETWEEN 0 AND 16. EACH VALUE IS USED
C TO FORM A 4 BY 4 ELEMENT WITH POINTS
C TURNED ON CORRESPONDING TO IT'S GREY LEVEL.
C THE ROW IS PLOTTED STARTING AT SCREEN ADDRESS
C       X = IX*4
C       Y = IY * 4
C
C PARAMETERS :
C       IX = THE X VALUE OF THE LOWER LEFTMOST
C            POINT OF THE ROW'S FIRST ELEMENT
C       IY = THE Y VALUE OF THE LLFT POINT
C       IZ = THE LOGICAL*1 ARRAY WITH THE GREY LEVELS
C       NZ = THE NUMBER OF POINTS IN IZ
C
C LIMITS :
C    0 < IX < 255       0 < IY < 190
C
C SUBROUTINES REQUIRED  LX,CKHX
C
      LOGICAL*1 IW(4),A(1),AF(256),AP(256),IZ(NZ),HY
      LOGICAL*1 HX,LAX,T,HXX
      COMMON /GRAP/LU,IXX,IYY,ISX,ISY,A
      EQUIVALENCE (A(1400),AF(1)),(A(1660),AP(1))
C CALCULATE SCOPE ADDRESSES
      KX=IX*4
      KXM=KX-1
      KY=IY*4-1
C CALCULATE LOW Y'S
      DO 10 I=1,4
      IW(I)=(KY+I .AND. "37)+"140
10    CONTINUE
      KYP=KY+1
      HY=("1740 .AND. KYP)/32+"040
      HX=("1740 .AND. KX)/32+"040
      LAX=("37 .AND. KX)+"100
C CALCULATE THE NUMBER OF FULL COLUMNS IN EACH
C ELEMENT AND THE NUMBER OF PTS IN THE PARTIALLY
C FILLED COLUMN
      DO 20 I=1,NZ
      IT=IZ(I) AND 255
      IFF=IT/4
      AP(I)=(IT/4.0-IFF)*4.0
      AF(I)=IFF
20    CONTINUE
C FILL UP THE OUTBUFFER WITH THE COMMANDS
C TO TURN OFF AND ON THE PEN AS IT TRAVELS
C ALONG THE 4 COLUMNS THAT MAKE UP THE ELEMENT
C COLUMN
      A(1)=0
      IN=2
      DO 50 I=1,4
```

```
      HXX=HX
      A(IN)=29
      A(IN+1)=HY
      IYC=IW(I)
      A(IN+2)=IYC
      A(IN+3)=HX
      A(IN+4)=LAX
      IN=IN+5
      LC=1
      IFL2=1
      IFL=0
      T=AF(1).EQ.0.AND.AP(1).EQ.0
      IF(T)IFL=1
      IF(T)LC=2
      IF(T)GO TO 35
C LOAD INITIAL ADDRESS OF THE CURRENT LINE
C INTO THE BUFFER THEN SEARCH FOR THE X WHICH
C MARKS THE END OF CONTINUOUS WRITING AND
C BEGIN WRITING DARK AND SO ON TO THE END
C OF THE COLUMN
30    IF(AF(LC).GT.0)GO TO 40
      IF(AP(LC).GT.0)IP=AP(LC)
      IF(AP(LC).EQ.0)GO TO 31
      IXC=KXM+(LC-1)*4+IP
      A(IN)=IYC
      IN=IN+1
      CALL CKHX(IXC,A,IN,HXX)
      AP(LC)=0
31    IP=0
      IFL=1
      GO TO 40
35    IF(AF(LC).GT.0)GO TO 37
      IF(AP(LC).GT.0)GO TO 37
      IFL2=IFL2+1
      IF(IFL2.EQ.NZ)GO TO 55
      GO TO 40
37    A(IN)=29
      A(IN+1)=IYC
      IN=IN+2
      CALL CKHX(KX+(LC-1)*4,A,IN,HXX)
      IFL=0
      GO TO 30
40    IF(IFL.NE.1)GO TO 45
      LC=LC+1
      IF(LC.GT.NZ)GO TO 49
      GO TO 35
45    AF(LC)=AF(LC)-1
      LC=LC+1
      IF(LC.GT.NZ)GO TO 49
      GO TO 30
49    IF(IFL.EQ.1)GO TO 50
      A(IN)=IYC
      IN=IN+1
      CALL CKHX(KXM+(LC-1)*4,A,IN,HXX)
50    CONTINUE
55    WRITE(LU,1)(A(I),I=1,IN-1)
```

```
1       FORMAT( '+', $, 124A1 )
        RETURN
        END
```

```
      SUBROUTINE LX(IX,A,IN)
C LX CONVERTS THE X COORDINATE TO ITS 2
C BYTE ADDRESS AND STORES THE BYTES IN
C A AS INDEXED BY IN
      LOGICAL*1 A(1)
      A(IN)=(IX.AND."1740)/32+"040
      A(IN+1)=(IX.AND."37)+"100
      RETURN
      END
      SUBROUTINE LY(IY,A,IN)
C LY CONVERTS IY TO ITS 2 BYTE ADDRESS
C AND STORES THE BYTES AS HI Y AND LO Y
C IN A AS INDEXED BY IN
      LOGICAL*1 A(1)
      A(IN)=(IY.AND."1740)/32+"040
      A(IN+1)=(IY.AND."37)+"140
      RETURN
      END
      SUBROUTINE CKHX(IX,A,IN,HX)
      LOGICAL*1 A(1),HX
C CKHX LOADS A HIGH X AND Y INTO THE APPROPRIATE
C POINT IN ARRAY A UNLESS THE NEW HX IS EQUAL
C TO THE PREVIOUS ONE IN WHICH CASE ONLY THE LX
C IS LOADED INTO A ASSUMING A LOWY IS ALREADY
C PRESENT LX IS LOADED INTO A(IN-1)
      CALL LX(IX,A,IN)
      IF(A(IN).EQ.HX)GO TO 10
      HX=A(IN)
      IN=IN+2
      RETURN
10    A(IN-1)=A(IN+1)
      RETURN
      END
      SUBROUTINE LOADB(IX,IY,A,IN)
C LOADB CONVERTS THE X AND Y COORDINATES TO
C A 4 BYTE ADDRESS USABLE BY THE TTY AND
C STORES THE 4 BYTES IN ARRAY A STARTING AT
C BYTE IN
      LOGICAL*1 A(1)
      A(IN)=(IY.AND."1740)/32+"040
      A(IN+1)=(IY.AND."37)+"140
      A(IN+2)=(IX.AND."1740)/32+"040
      A(IN+3)=(IX.AND."37)+"100
      RETURN
      END
      SUBROUTINE INIT(LU)
C ERASES SCREEN HOMES CURSOR
      I="006033
      WRITE(LU,1)I
1     FORMAT('$',A2)
      RETURN
      END
```

```
      SUBROUTINE GRID(NL)
C GRID WRITES OUT NL EVENLY SPACED HORIZONTAL
C GRID LINES FILLING THE GRAPHICS WINDOW
C WITH EXTENSIONS PASSING THE LEFT SIDE OF
C THE WINDOW PROVIDING THE TIC MARKS
C
C PARAMETERS :
C       NL = THE NUMBER OF DIVISIONS INSIDE THE
C            WINDOW. NL+1 LINES ARE DRAWN
C
C SUBROUTINES REQUIRED :
C       FROM GRAPH - LX,LY
C
      LOGICAL*1 A(1),IB(4)
      COMMON /GRAP/LU,IX,IY,ISX,ISY,A
      ID=ISY/NL
      JX=IX+ISX
      KX=IX-20
      A(1)=29
      CALL LX(JX,IB,1)
      CALL LX(KX,IB,3)
      JY=IY
      DO 10 I=1,NL+1
      IN=(I-1)*8+2
      CALL LY(JY,A,IN)
      A(IN+2)=IB(1)
      A(IN+3)=IB(2)
      A(IN+4)=A(IN+1)
      A(IN+5)=IB(3)
      A(IN+6)=IB(4)
      A(IN+7)=29
      JY=JY+ID
10    CONTINUE
      WRITE(LU,1)(A(I),I=1,8*NL+1)
1     FORMAT('+',$,180A1)
      RETURN
      END
      SUBROUTINE FRAME()
C FRAME WRITES A BOX AROUND THE WINDOW
C AS DELINEATED BY THE COMMON VARIABLES
      LOGICAL*1 A(1)
      COMMON /GRAP/LU,IX,IY,ISX,ISY,A
      ILX=IX
      ILY=IY
      CALL LOADB(ILX,ILY,A,2)
      ILX=ILX+ISX
      CALL LOADB(ILX,ILY,A,6)
      ILY=ILY+ISY
      CALL LOADB(ILX,ILY,A,10)
      ILX=IX
      CALL LOADB(ILX,ILY,A,14)
      ILY=IY
      CALL LOADB(ILX,ILY,A,18)
      WRITE(LU,1)(A(I),I=1,21)
1     FORMAT('+',$,21A1)
      RETURN
      END
```

```
      SUBROUTINE AXIS(XV,YV,NX,NY)
C AXIS WRITES OUT THE VECTOR OF Y AND X
C VALUES,GIVEN IN XV AND YV,EVENLY DISTRIBUTED
C ALONG THE WINDOW AS SET FORTH IN THE COMMON VARIABLES
C
C PARAMETERS :
C       XV = THE VALUES TO BE WRITTEN ALONG THE X AXIS
C       YV = THE VALUES TO BE WRITTEN ALONG THE Y AXIS
C       NX = THE NUMBER OF X VALUES
C       NY = THE NUMBER OF Y VALUES
C
C LIMITS :
C       THE X OR Y VALUES MUST NOT EXCEED 15
C
C SUBROUTINES REQUIRED LOADB
C
      LOGICAL*1 A(1)
      INTEGER*2 XV(NX),YV(NY)
      COMMON /GRAP/LU,IX,IY,ISX,ISY,A
      A(1)=29
      A(6)=31
      ND=ISY/(NY-1)
      ID=ISX/(NX-1)
      JY=IY
      JX=IX-32
C WRITE OUT THE VALUES ALONG THE Y AXIS
      DO 10 I=1,NY
      CALL LOADB(0,JY,A,2)
      WRITE(LU,1)(A(J),J=1,6),YV(I)
1     FORMAT('+',6A1,I5)
      JY=JY+ND
10    CONTINUE
C WRITE OUT THE X AXIS VALUES
      DO 20 I=1,NX
      CALL LOADB(JX,50,A,2)
      WRITE(LU,1)(A(J),J=1,6),XV(I)
      JX=JX+ID
20    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TICX(LT,NM,NS)
      LOGICAL*1 A(1),Y(4),LYY,HX,HY
      COMMON /GRAP/LU,IX,IY,ISX,ISY,A
C TICX PLOTS NM-1 MAJOR TIC MARKS OF LENGTH
C LT AND NS-1 MINOR TIC MARKS OF LENGTH LT/2 IN
C BETWEEN EACH PAIR OF MAJOR TIC MARKS ON THE X
C AXIS
C
C SUBROUTINES REQUIRED - LOADB,LY,LX
C
      LTS=LT/2
      NI=ISX/NM
      IF((FLOAT(ISX)/NM-NI).GE..5)NI=NI+1
      NSI=NI/NS
      CALL LY(IY-LT,Y,1)
      CALL LY(IY-LTS,Y,3)
      A(1)=29
      CALL LOADB(IX,IY,A,2)
      HY=A(2)
      LYY=A(3)
      HX=A(4)
      IN=6
      NT=NM*NS+1
      JM=0
      DO 20 I=1,NT
      IF((I-1)/NS*NS.EQ.(I-1))GO TO 10
      KX=KX+NSI
      A(IN)=29
      A(IN+1)=HY
      A(IN+2)=LYY
      CALL LX(KX,A,IN+3)
      IN=IN+5
      A(IN)=Y(3)
      A(IN+1)=Y(4)
      A(IN+2)=A(IN-1)
      IN=IN+3
      GO TO 20
10    KX=IX+JM*NI
      A(IN)=29
      A(IN+1)=HY
      A(IN+2)=LYY
      CALL LX(KX,A,IN+3)
      IN=IN+5
      A(IN)=Y(1)
      A(IN+1)=Y(2)
      A(IN+2)=A(IN-1)
      IN=IN+3
      JM=JM+1
20    CONTINUE
      WRITE(LU,1)(A(I),I=1,IN-1)
1     FORMAT('+',$,124A1)
      RETURN
      END
```

UNIVAC 1108 FORTRAN PROGRAMS
AND SUBROUTINES

```
1016*TEMP(1).IMAGAN
      1       C
      2       C
      3       C     THIS PROGRAM GENERATES AN IMAGE FROM THE BOUNDARY
      4       C     POINTS, WHILE ADDING A NOISY FIELD.  THE PICTURE IS
      5       C     THRESHOLDED AND PRINTED IN A BINARY FORM, FINALLY
      6       C     THE BOUNDARY OF THE NOISY IMAGE IS TRACED AND
      7       C     WRITTEN ON UNIT 8.
      8       C
      9       C     INPUT REQUIRED:
     10       C        1) BOUNDARY POINTS, IDENTIFIER AND NUMBER OF
     11       C           POINTS ON FIRST RECORD, (A6,I4) FORMAT,
     12       C           BOUNDARY COORDINATES ON SUBSEQUENT RECORDS
     13       C           (2F5.2) FORMAT, UNIT 7
     14       C        2) GENERATED IMAGE NOISE, AS OUTPUTTED BY NOISE
     15       C           GENERATING PROGRAM, UNIT 9
     16       C        3) STARTING POINT FOR BOUNDARY TRACE AND RELATIVE
     17       C           DIRECTION FOR TRACING RAY, FREE FORMAT, UNIT 5
     18       C
     19       C     OUTPUT:
     20       C        1) IMAGE WITH NOISE
     21       C        2) CONTOUR OF NOISY IMAGE
     22       C        3) BOUNDARY POINTS OF NOISY IMAGE, SAME FORMAT AS
     23       C           INPUT, UNIT 8
     24       C
     25       C     REQUIRES SUBROUTINES INBND, SCALE, DEC, BNDTRC, SORT,
     26       C     & PRNTBD
     27       C
     28       C
     29              DIMENSION IMAG(100),LINE(100),BOUND(300),IBND(100)
     30             1,PICT(50,50),IS1(100)
     31              REAL IMAG,NAME
     32              DATA NY/3HYES/
     33       C
     34       10     CONTINUE
     35       C
     36                 DO 15 J=1,100
     37                 IBND(J)=0
     38                 IS1(J)=0
     39       15        CONTINUE
     40       C
     41              CALL INBND(NAME,BOUND,ILIM)
     42       C
     43              READ(9,30) N,M,SIGMA,ACOR
     44       30     FORMAT(2I3,10X,2F10.5)
     45              WRITE(6,40) SIGMA,ACOR
     46       40     FORMAT(/' *** FIRST ORDER MARKOV NOISE WITH'/
     47             1'     STANDARD DEVIATION:',F6.2,'     CORRELATION '
     48             2,'COEFFICIENT:',F6.2,/)
     49       C
     50              I=1
     51              CALL SCALE(BOUND,ILIM,N,M)
```

```
52      C
53            DO 100 J=1,N
54            READ(9,50) (IMAG(K),K=1,M)
55      50    FORMAT(10F10.5)
56            CALL DEC(BOUND,IMAG,LINE,I,ILIM,N,M,IBND,PICT,J,IS1)
57      100   CONTINUE
58      C
59          READ(5,300) (BOUND(J),J=1,2),IX,IY
60      300 FORMAT()
61          CALL BNDTRC(PICT,BOUND,IX,IY,M,ILIM)
62          IL2=ILIM+ILIM
63          WRITE(8,600) NAME,ILIM,(BOUND(J),J=1,IL2)
64      600 FORMAT(A6,1X,I3,/,(2F5.2))
65          ENDFILE 8
66          CALL SORT(BOUND,ILIM)
67          WRITE(6,999) NAME
68      999 FORMAT('1*** EDGE CONTOUR FOR NOISY ',A6/)
69          CALL PRNTBD(BOUND,ILIM,N,M,LINE)
70      C
71          STOP
72          END
```

```
1016*TEMP(1).INBND
      1              SUBROUTINE INBND(NAME,BOUND,ILIM)
      2        C
      3        C
      4        C     ROUTINE TO READ BOUNDARY DATA FROM UNIT 7, FORMAT
      5        C     AS DESCRIBED IN MAIN PROGRAM
      6        C
      7        C     NAME....IDENTIFIER
      8        C     BOUND...ARRAY OF BOUNDARY POINTS
      9        C     ILIM....NUMBER OF BOUNDARY POINTS
     10        C
     11        C     REQUIRES SUBROUTINES CLBND & SORT
     12        C
     13        C
     14              DIMENSION BOUND(1)
     15        C
     16              READ(7,10) NAME,ILIM
     17        10    FORMAT(A6,1X,I3)
     18        C
     19        C
     20              II=ILIM+ILIM
     21              READ(7,40) (BOUND(J),J=1,II)
     22        40    FORMAT(2F5.2)
     23        C
     24              CALL CLBND(BOUND,ILIM)
     25              CALL SORT(BOUND,ILIM)
     26        C
     27              RETURN
     28              END
```

```
1016*TEMP(1).CLBND
     1              SUBROUTINE CLBND(BND,L)
     2        C
     3        C
     4        C     ROUTINE TO PREPARE BOUNDARY POINT ARRAY
     5        C     FOR DEC SUBROUTINE
     6        C
     7        C     BND.....ARRAY OF BOUNDARY POINTS
     8        C     L.......NUMBER OF BOUNDARY POINTS
     9        C
    10        C     NO EXTERNAL SUBROUTINES REQUIRED
    11        C
    12        C
    13              DIMENSION BND(1)
    14        C
    15              L1=L+L-1
    16              X1=BND(L1)
    17              L1=L1+1
    18              Y1=BND(L1)
    19              X2=BND(1)
    20              Y2=BND(2)
    21              I=2
    22              L1=L-1
    23        C
    24                 DO 100 J=1,L1
    25                 J1=J+J-1
    26                 I=I+1
    27                 X3=BND(I)
    28                 I=I+1
    29                 Y3=BND(I)
    30                 A1=ANG(X1,Y1,X2,Y2,X3,Y3)
    31                 Y4=Y2+1
    32                 A2=ANG(X1,Y1,X2,Y2,X2,Y4)
    33                 IF(A1.GT.A2) BND(J1)=X2+1000
    34                 X1=X2
    35                 Y1=Y2
    36                 X2=X3
    37                 Y2=Y3
    38        100      CONTINUE
    39        C
    40              J1=L+L-1
    41              X3=BND(1)
    42              IF(X3.GT.995.) X3=X3-1000
    43              Y3=BND(2)
    44              A1=ANG(X1,Y1,X2,Y2,X3,Y3)
    45              Y4=Y2+1
    46              A2=ANG(X1,Y1,X2,Y2,X2,Y4)
    47              IF(A1.GT.A2) BND(J1)=X2+1000
    48        C
    49              RETURN
    50              END
```

```
1016*TEMP(1).SORT
      1             SUBROUTINE SORT(BOUND,ILIM)
      2       C
      3       C
      4       C     ROUTINE TO SORT BOUNDARY POINTS SUCH THAT Y VALUES
      5       C     (EVEN INDICES) ARE IN DECREASING ORDER
      6       C
      7       C     BOUND...ARRAY OF BOUNDARY POINTS (TWO ELEMENTS/POINT)
      8       C     ILIM....NUMBER OF BOUNDARY POINTS
      9       C
     10       C     NO EXTERNAL SUBROUTINES REQUIRED
     11       C
     12       C
     13             DIMENSION BOUND(1)
     14       C
     15             ILIM1=ILIM-1
     16       C
     17                DO 100 J=1,ILIM1
     18                JJ=J+J
     19                JJ1=JJ-1
     20                J1=J+1
     21                B1=BOUND(JJ)
     22       C
     23                   DO 80 K=J1,ILIM
     24                   KK=K+K
     25                   B2=BOUND(KK)
     26                   IF(B1-B2) 80,80,40
     27       40          CONTINUE
     28                   BOUND(KK)=B1
     29                   B1=B2
     30                   BOUND(JJ)=B2
     31                   B2=BOUND(JJ1)
     32                   KK1=KK-1
     33                   BOUND(JJ1)=BOUND(KK1)
     34                   BOUND(KK1)=B2
     35       80          CONTINUE
     36       C
     37       100      CONTINUE
     38       C
     39             RETURN
     40             END
```

```
1016*TEMP(1).SCALE
      1              SUBROUTINE SCALE(BOUND,ILIM,N,M)
      2        C
      3        C
      4        C     ROUTINE TO CENTER IMAGE IN NXM PICTURE
      5        C
      6        C     BOUND...BOUNDARY POINT ARRAY
      7        C     ILIM....NUMBER OF BOUNDARY POINTS
      8        C     N.......NUMBER OF LINES IN PICTURE
      9        C     M.......NUMBER OF COLUMNS IN PICTURE
     10        C
     11        C     NO EXTERNAL SUBROUTINES REQUIRED
     12        C
     13        C
     14              DIMENSION BOUND(1)
     15        C
     16              XSCALE=M/2.0-50
     17              YSCALE=N/2.0-50
     18              K=0
     19        C
     20                 DO 20 J=1,ILIM
     21                 K=K+1
     22                 BOUND(K)=XSCALE+BOUND(K)
     23                 K=K+1
     24                 BOUND(K)=YSCALE+BOUND(K)
     25        20       CONTINUE
     26        C
     27              RETURN
     28              END
```

```
1016*TEMP(1).DEC
      1              SUBROUTINE DEC(BOUND,IMAG,LINE,I,ILIM,N,M,IBND,
      2             1PICT,NN,IS1)
      3       C
      4       C
      5       C     ROUTINE TO THRESHOLD AND PRINT A LINE OF BINARY
      6       C     PICTURE, A (0,1) PICTURE IS STORED IN PICT
      7       C
      8       C     BOUND...ARRAY OF BOUNDARY POINTS
      9       C     IMAG....ARRAY CONTAINING LINE OF NOISE
     10       C     LINE....WORKING ARRAY WITH AT LEAST M ELEMENTS
     11       C     I.......(NUMBER OF BOUNDARY POINTS PREVIOUSLY USED)+1
     12       C             SAVE BETWEEN CALLS
     13       C     ILIM....NUMBER OF BOUNDARY POINTS
     14       C     N.......NUMBER OF LINES IN PICTURE
     15       C     M.......NUMBER OF COLUMNS IN PICTURE
     16       C     IBND....WORK ARRAY WITH AT LEAST M ELEMENTS, SAVE
     17       C             BETWEEN CALLS
     18       C     PICT....ARRAY FOR RETURNING BINARY PICTURE
     19       C     NN......CURRENT LINE NUMBER
     20       C     IS1.....WORK ARRAY WITH AT LEAST M ELEMENTS, SAVE
     21       C             BETWEEN CALLS
     22       C
     23       C     NO EXTERNAL SUBROUTINES REQUIRED
     24       C
     25       C
     26              DIMENSION BOUND(1),IMAG(1),LINE(1),IBND(1),
     27             1PICT(1),IS1(1)
     28              REAL IMAG
     29              INTEGER BLNK,ONE,PICT
     30              DATA BLNK/1H /,ONE/1H*/
     31              NR=M*(NN-1)
     32       C
     33       10     CONTINUE
     34              IF(I-ILIM) 20,20,100
     35       20     CONTINUE
     36              II=I+I
     37              L=BOUND(II)
     38              IF(L-NN) 100,30,100
     39       30     CONTINUE
     40              II1=II-1
     41              L1=BOUND(II1)
     42              IF(L1.LT.995) GO TO 40
     43              L1=L1-1000
     44              IS1(L1)=1
     45              GO TO 50
     46       40     CONTINUE
     47              IS1(L1)=0
     48       50     CONTINUE
     49              IBND(L1)=1
     50              I=I+1
     51              GO TO 10
```

```
      C
100   CONTINUE
      C
          DO 200 J=1,M
          NNJ=NR+J
          IX=(IBND(J)+IS1(J)+1.5)/2.0
          IBND(J)=0
          IF(IMAG(J)+IX-.5) 160,160,170
160       CONTINUE
          LINE(J)=BLNK
          PICT(NNJ)=0
          GO TO 200
170       CONTINUE
          LINE(J)=ONE
          PICT(NNJ)=1
200       CONTINUE
      C
          WRITE(6,300) (LINE(J),J=1,M)
300   FORMAT(5X,100A1)
          RETURN
          END
```

```
1016*TEMP(1).BNDTRC
      1             SUBROUTINE BNDTRC(PICT,BND,IX,IY,N,LIM)
      2       C
      3       C
      4       C     ROUTINE TO TRACE IMAGE BOUNDARY
      5       C
      6       C     PICT....(0,1) BINARY PICTURE, REPRESENTING IMAGE
      7       C     BND.....RETURNED IMAGE POINTS
      8       C     BND(1)..STARTING X COORDINATE, SUPPLIED BY CALLING
      9       C             ROUTINE, NUMBER OF COLUMNS TO LEFT OF RIGHT
     10       C             EDGE OF PICTURE
     11       C     BND(2)..STARTING Y COORDINATE, NUMBER OF LINES FROM
     12       C             TOP OF PICTURE
     13       C     IX......STARTING POSITION OF SWEEP RAY, RELATIVE TO
     14       C             TO STARTING X COORDINATE
     15       C     IY......STARTING POSITION OF SWEEP RAY, RELATIVE TO
     16       C             TO STARTING Y COORDINATE
     17       C     N.......NUMBER OF ROWS IN PICTURE
     18       C     LIM.....NUMBER OF BOUNDARY POINTS FOUND
     19       C
     20       C     NO EXTERNAL SUBROUTINES REQUIRED
     21       C
     22       C
     23             DIMENSION PICT(N,1),BND(1)
     24             INTEGER X1,Y1,X,Y,PICT,XT,YT
     25       C
     26             X1=BND(1)
     27             Y1=BND(2)
     28             X=X1
     29             Y=Y1
     30             I=3
     31       C
     32       10       CONTINUE
     33                IXL=IX
     34                IYL=IY
     35                IF(IX*IY.LE.0.AND.IY.NE.0) GO TO 20
     36                IY=IY-IX
     37                GO TO 30
     38       20       CONTINUE
     39                IX=IY+IX
     40       30       CONTINUE
```

```
      C
            XT=X+IX
            YT=Y-IY
            IF(XT.LE.0.OR.YT.LE.0) GO TO 10
            IF(PICT(XT,YT)) 10,10,40
40          CONTINUE
            IF(X1.EQ.XT.AND.YT.EQ.Y1) GO TO 50
            BND(I)=XT
            I=I+1
            BND(I)=YT
            I=I+1
            X=XT
            Y=YT
            IX=IXL-IX
            IY=IYL-IY
            GO TO 10
      C
50    CONTINUE
      LIM=(I-1)/2
      RETURN
      END
```

```
1016*TEMP(1).PRNTBD
      1              SUBROUTINE PRNTBD(BND,LIM,N,M,LINE)
      2        C
      3        C
      4        C     ROUTINE TO PRINT BOUNDARY CONTOUR
      5        C
      6        C     BND.....ARRAY OF BOUNDARY POINTS
      7        C     LIM.....NUMBER OF BOUNDARY POINTS
      8        C     N.......NUMBER OF LINES IN PICTURE
      9        C     M.......NUMBER OF CHARACTERS PER LINE
     10        C     LINE....WORKING ARRAY OF NOT LESS THAN M ELEMENTS
     11        C
     12        C     NO EXTERNAL SUBROUTINES REQUIRED
     13        C
     14        C
     15              DIMENSION BND(1),LINE(1)
     16              INTEGER STAR,BLNK
     17              DATA STAR/1H*/,BLNK/1H /
     18        C
     19              LL=LIM+LIM
     20              I=2
     21        C
     22                 DO 100 J=1,N
     23        C
     24                    DO 20 K=1,M
     25                    LINE(K)=BLNK
     26        20          CONTINUE
     27        C
     28        30       CONTINUE
     29                 IF(I.GT.LL) GO TO 70
     30                 L1=BND(I)
     31                 IF(L1.NE.J) GO TO 70
     32                 I1=I-1
     33                 L1=BND(I1)
     34                 LINE(L1)=STAR
     35                 I=I+2
     36                 GO TO 30
     37        70       CONTINUE
     38                 WRITE(6,80) (LINE(K),K=1,M)
     39        80       FORMAT(5X,100A1)
     40        100      CONTINUE
     41        C
     42              RETURN
     43              END
```

```
1016*TEMP(1).PROG2
      1       C
      2       C
      3       C     THIS PROGRAM PERFORMS VARIOUS ANALYSIS OF IMAGE DATA
      4       C     IN THE FORM OF THE ANGLE PULSE FUNCTION EXTRACTED FROM
      5       C     THE IMAGE OUTLINE
      6       C
      7       C     INPUT REQUIRED:
      8       C        1) EDGE POINTS ON UNIT 8 WITH IDENTIFIER & NUMBER OF
      9       C           POINTS ON FIRST RECORD IN (A6,I4) FORMAT, EDGE POINTS
     10       C           ON REMAINING RECORDS IN (2F5.2) FORMAT
     11       C        2) NUMBER OF INCREMENTS, UNIT 5
     12       C        3) NUMBER OF FREQUENCIES FOR FOURIER COEFFICIENTS,
     13       C           UNIT 5
     14       C        4) INITIAL FILTER WIDTH, UNIT 5
     15       C        5) FILTER WIDTH INCREMENT
     16       C
     17       C     MISC. AT EACH OUTPUT STEP INPUT 'NO' IF NO OUTPUT IS
     18       C           DESIRED, OTHERWISE CARRIAGE RETURN
     19       C
     20       C     OUTPUT:
     21       C        1) EDGE POINTS
     22       C        2) THETA(L)
     23       C        3) THETA PRIME(L)
     24       C        4) THETA PRIME AFTER GAUSSIAN FILTER
     25       C        5) FOURIER COEFFICIENTS OF THETA PRIME
     26       C
     27       C     REQUIRES SUBROUTINES COMP1, FTRANS, NORM, PLOT, & GFILT
     28       C
     29       C
     30             DIMENSION EDGE(300),THETA(300),THPRIM(300),THTRAN(300)
     31            1,Z(300)
     32             REAL NAME,NORM
     33             DATA ST/6HSTOP  /,QA/6HNO    /
     34       C
     35             WRITE(6,10)
     36       10    FORMAT(' *** NUMBER OF INCREMENTS, FREQUENCIES, ', & D***/)
     37             READ(5,20) N,M,','D
     38       20    FORMAT()
     39       C
     40       30    CONTINUE
     41             READ(8,35) NAME,L
     42       35    FORMAT(A6,1X,I3)
     43             IF(NAME.EQ.ST) GO TO 300
     44             IF(N.LE.0) N=L
     45             I1=0
     46       C
     47                DO 50 J=1,L
     48                I=I1+1
     49                I1=I+1
     50                READ(8,40) EDGE(I),EDGE(I1)
     51       40       FORMAT(2F5.2)
     52       50       CONTINUE
```

```
      C
            CALL COMP1(L,EDGE,N,THETA,THPRIM)
            CALL FTRANS(N,THPRIM,M,THTRAN)
            THNRM=NORM(THETA,N)
            THPNRM=NORM(THPRIM,N)
      C
            WRITE(6,100) NAME,L,N
      100   FORMAT('1*** ',A6,' *** LENGTH: ',I3,5X,'NO. INC: ',I3)
      C
            READ(5,400) QS
            IF(QA.EQ.QS) GO TO 115
            LL=L+1
            WRITE(6,110) (EDGE(J),J=1,LL)
      110   FORMAT('0*** EDGE POINTS'/(1X,5('(',F5.2,',',F5.2,')',2X)))
      C
      115   CONTINUE
            READ(5,400) QS
            IF(QS.EQ.QA) GO TO 125
            WRITE(6,120) (THETA(J),J=1,N)
      120   FORMAT('0*** THETA'/(10F9.2))
            WRITE(6,122)
      122   FORMAT(/'0*** THETA VS. L'/)
            CALL PLOT(THETA,N,N,26)
      C
      125   CONTINUE
      C
               DO 700 NN=1,N
               THETA(NN)=THPRIM(NN)
      700      CONTINUE
      C
            READ(5,400) QS
            IF(QS.EQ.QA) GO TO 1130
            WRITE(6,130) (THPRIM(J),J=1,N)
      130   FORMAT('0*** THETAPRIME'/(10F9.2))
            WRITE(6,131)
      131   FORMAT(/'0*** THETAPRIME VS. L'/)
            CALL PLOT(THPRIM,N,N,26)
      C
      1130  CONTINUE
            READ(5,400) QS
            IF(QS.EQ.QA) GO TO 138
      C
               DO 800 NW=1,5
               READ(5,400) QS
               IF(QS.EQ.QA) GO TO 800
               CALL GFILT(THETA,N,L,W,Z)
               WRITE(6,132) W,(Z(J),J=1,N)
      132      FORMAT('0*** FILTERED THETAPRIME (GAUSS), W = ',F5.2
           1   /(10F9.2))
               WRITE(6,134)
      134      FORMAT(/'0*** FILTERED THETAPRIME VS. L'/)
               CALL PLOT(Z,N,N,26)
               W=W+DW
      800      CONTINUE
```

```
      C
  138 CONTINUE
      READ(5,400) QS
      IF(QS.EQ.QA) GO TO 300
      C
      C=ABS(THTRAN(1))
      WRITE(6,140) THTRAN(1),C
  140 FORMAT('0*** FOURIER COEFFICIENTS'/10X,'W',10X,'A',10X,'B',
     110X,'C'/10X,'0',5X,F11.4,11X,F11.4)
      C
          DO 145 J=1,M
          J1=J+J
          J2=J1+1
          A=THTRAN(J1)
          B=THTRAN(J2)
          C=SQRT(A*A+B*B)
          WRITE(6,142) J,A,B,C
  142     FORMAT(9X,I2,5X,3F11.4)
  145     CONTINUE
      C
      WRITE(6,150) THNRM,THPNRM
  150 FORMAT('0*** THNRM: ',F7.2,'   THPNRM: ',F7.2)
      C
  300 CONTINUE
      STOP
  400 FORMAT(A6)
      END
```

```
1016*TEMP(1).COMP1
      1              SUBROUTINE COMP1(L,EDGE,N,THETA,THPRIM)
      2        C
      3        C
      4        C          ROUTINE TO COMPUTE BOUNDARY ANGLE AND ITS
      5        C          DERIVATIVE AS A FUNCTION OF THE BOUNDARY
      6        C          LENGTH, FROM THE COORDINATES OF THE BOUNDARY
      7        C
      8        C          L.......LENGTH OF BOUNDARY
      9        C          EDGE....BOUNDARY COORDINATES IN CLOCKWISE
     10        C                  ORDER, EX. X(10),Y(10) LOCATED AT
     11        C                  EDGE(19),(20)
     12        C          N.......NUMBER OF EDGE POINTS
     13        C          THETA...COMPUTED ANGLE FUNCTION
     14        C          THPRIM..COMPUTED DERIVATIVE OF THETA
     15        C
     16        C          REQUIRES STANDARD FORTRAN SUBROUTINES ONLY
     17        C
     18        C
     19              DIMENSION EDGE(1),THETA(1),THPRIM(1)
     20              A=180/3.1415926
     21              I=2
     22              X1=EDGE(1)
     23              Y1=EDGE(2)
     24        C
     25              L1=L-1
     26        C
     27                 DO 50 J=1,L1
     28                 I=I+1
     29                 X2=EDGE(I)
     30                 I=I+1
     31                 Y2=EDGE(I)
     32                 YD=Y2-Y1
     33                 XD=X2-X1
     34                 THT=ACOS(XD/SQRT(XD*XD+YD*YD))
     35                 IF(YD.GT.0) THT=-THT
     36                 THPRIM(J)=THT*A
     37                 X1=X2
     38                 Y1=Y2
     39        50       CONTINUE
     40        C
     41              X2=EDGE(1)
     42              Y2=EDGE(2)
     43              YD=Y2-Y1
     44              XD=X2-X1
     45              THT=ACOS(XD/SQRT(XD*XD+YD*YD))
     46              IF(YD.GT.0) THT=-THT
     47              THPRIM(L)=THT*A
```

```
      C
      C     NORMALIZE THETA
      C
            S=FLOAT(L)/N
            SJ=-S
      C
               DO 100 J=1,N
               SJ=S+SJ
               I=SJ+1
               THETA(J)=THPRIM(I)
  100          CONTINUE
      C
            THT=THETA(1)-THETA(N)
            IF(THT.GE.179.999) THT=THT-360
            IF(THT.LT.-180.001) THT=THT+360
            THPRIM(1)=THT
      C
               DO 150 J=2,N
               I=J-1
               THT=THETA(J)-THETA(I)
               IF(THT.GE.179.999) THT=THT-360
               IF(THT.LT.-180.001) THT=THT+360
               THPRIM(J)=THT
  150          CONTINUE
      C
            RETURN
            END
```

```
1016*TEMP(1).FTRANS
      1             SUBROUTINE FTRANS(N,F,M,FT)
      2       C
      3       C
      4       C     ROUTINE TO CALCULATE THE FOURIER COEFFICIENTS
      5       C     OF F, ASSUMING AN INTERVAL LENGTH OF TWO PI
      6       C
      7       C     N.......NUMBER OF INCREMENTS FOR F
      8       C     F.......ARRAY OF FUNCTION VALUES OF F AT EQUAL INCREMENTS
      9       C     M.......NUMBER OF FREQUENCIES TO BE CALCULATED
     10       C     FT......COEFFICIENT VALUES RETURNED, IN ORDER OF INCREASING
     11       C               FREQUENCY (COS,SIN)
     12       C
     13       C     REQUIRES STANDARD FORTRAN SUBROUTINES ONLY
     14       C
     15       C
     16             DIMENSION F(1),FT(1)
     17             W=6.28318531/N
     18             FT1=0
     19       C
     20                DO 50 J=1,N
     21                FT1=FT1+F(J)
     22       50       CONTINUE
     23       C
     24             FT(1)=FT1/6.28318531
     25             WT=0
     26       C
     27                DO 100 J=1,M
     28                FT1=0
     29                FT2=0
     30                WT=WT+W
     31                WTT=0
     32       C
     33                   DO 70 I=1,N
     34                   WTT=WT+WTT
     35                   F1=F(I)
     36                   FT1=FT1+COS(WTT)*F1
     37                   FT2=FT2+SIN(WTT)*F1
     38       70          CONTINUE
     39       C
     40                JJ=J+J
     41                FT(JJ)=FT1/3.141592654
     42                JJ=JJ+1
     43                FT(JJ)=FT2/3.141592654
     44       100      CONTINUE
     45       C
     46             RETURN
     47             END
```

```
1016*TEMP(1).NORM
      1              FUNCTION NORM(F,N)
      2        C
      3        C
      4        C     ROUTINE TO CALCULATE L1 NORM OF F FOR AN INTERVAL LENGTH
      5        C     OF ONE
      6        C
      7        C     F.......ARRAY OF FUNCTION VALUES
      8        C     N.......NUMBER OF COMPONENTS OF F
      9        C
     10        C     REQUIRES STANDARD FORTRAN ROUTINES ONLY
     11        C
     12        C
     13              REAL NORM
     14              DIMENSION F(1)
     15              NORM=0
     16        C
     17                 DO 10 J=1,N
     18                 NORM=NORM+ABS(F(J))
     19        10       CONTINUE
     20        C
     21              NORM=NORM/N
     22              RETURN
     23              END
```

```
1016*TEMP(1).PLOT
      1              SUBROUTINE PLOT(Y,IY,N,M)
      2        C
      3        C
      4        C     ROUTINE TO PLOT VECTOR OF Y VALUES ON LINE PRINTER
      5        C
      6        C     Y.......ARRAY OF VALUES TO BE PLOTTED
      7        C     N.......NUMBER OF VALUES IN Y
      8        C     M.......NUMBER OF LINES TO BE USED FOR Y AXIS
      9        C
     10        C     REQUIRES ROUTINES SRT & PLT1
     11        C
     12        C
     13              DIMENSION X(250),Y(1),FORM(4)
     14              DATA FORM/6H(12X, ,6H10(2X,,6HF5.1, ,6H3X))  /
     15        C
     16              X(1)=0
     17              XT=0
     18        C
     19                 DO 20 J=2,IY
     20                 XT=XT+1
     21                 X(J)=XT
     22        20       CONTINUE
     23        C
     24              CALL SRT(X,Y,IY,XMIN,XMAX)
     25        C
     26              XS=(N-1)/(XMAX-XMIN)
     27              YMIN=Y(IY)
     28              YS=(M-1)/(Y(1)-YMIN)
     29        C
     30              CALL PLT1(X,Y,IY,N,M,XS,YS,XMIN,YMIN,
     31             110,5,1H*,FORM)
     32              RETURN
     33              END
```

```
1016*TEMP(1).SRT
     1              SUBROUTINE SRT(X,Y,IX,XMIN,XMAX)
     2        C
     3        C
     4        C     ROUTINE TO SORT DATA AND FIND MAXIMUM & MINIMUM X VALUES
     5        C     FOR PLT1
     6        C
     7        C     X.......X VECTOR RETURNED IN ORDER CORRESPONDING TO Y VECTOF
     8        C     Y.......Y VECTOR RETURNED IN DECREASING ORDER
     9        C     IX......NUMBER OF COMPONENTS IN X AND Y
    10        C     XMIN....MINIMUM X VALUE
    11        C     XMAX....MAXIMUM X VALUE
    12        C
    13        C     NO SUBROUTINES REQUIRED
    14        C
    15        C
    16              DIMENSION X(1),Y(1)
    17        C
    18              XMIN=X(1)
    19              XMAX=XMIN
    20              IX1=IX-1
    21        C
    22                 DO 100 J=1,IX1
    23                 J1=J+1
    24                 XT=X(J)
    25                 YT=Y(J)
    26        C
    27                    DO 50 JJ=J1,IX
    28                    YTT=Y(JJ)
    29                    IF(YTT.LT.YT) GO TO 50
    30                    Y(JJ)=YT
    31                    YT=YTT
    32                    YTT=X(JJ)
    33                    X(JJ)=XT
    34                    XT=YTT
    35        50          CONTINUE
    36        C
    37                 X(J)=XT
    38                 Y(J)=YT
    39                 IF(XT.GT.XMAX) GO TO 60
    40                 IF(XT.LT.XMIN) XMIN=XT
    41                 GO TO 100
    42        60       CONTINUE
    43                 XMAX=XT
    44        100      CONTINUE
    45        C
    46              XT=X(IX)
    47              IF(XT.GT.XMAX) XMAX=XT
    48              IF(XT.LT.XMIN) XMIN=XT
    49              RETURN
    50              END
```

```
1016*TEMP(1).PLT1
     1              SUBROUTINE PLT1(X,Y,IX,N,M,XS,YS,XMIN,YMIN,
     2             1INCX,INCY,CHAR,F)
     3       C
     4       C
     5       C      ROUTINE FOR LINE PLOT OF DATA POINTS GIVEN IN
     6       C      X AND Y ARRAYS, AS RETURNED BY SRT SUBROUTINE
     7       C
     8       C      X.......HORIZONTAL LOCATION VALUES
     9       C      Y.......VERTICAL LOCATION VALUES IN DECREASING ORDER
    10       C      IX......NUMBER OF POINTS TO BE PLOTTED
    11       C      N.......NUMBER OF CHARACTERS IN GRAPH PER LINE
    12       C      M.......NUMBER OF LINES IN GRAPH
    13       C      XS......1/(INCREMENT IN X CORRESPONDING TO 1 CHARACTER)
    14       C      YS......1/(INCREMENT IN Y CORRESPONDING TO 1 LINE)
    15       C      XMIN....MINIMUM X VALUE
    16       C      YMIN....MINIMUM Y VALUE
    17       C      INCX....NUMBER OF CHARACTERS BETWEEN VALUE LABELS ON X AXIS
    18       C      INCY....NUMBER OF LINES BETWEEN VALUE LABELS ON Y AXIS
    19       C      CHAR....CHARACTER TO BE USED FOR PLOTTING
    20       C      F.......FORMAT ARRAY FOR X AXIS VALUE LABELS LINE
    21       C
    22       C      NO EXTERNAL SUBROUTINES REQUIRED
    23       C
    24       C
    25              DIMENSION X(1),Y(1),GRAPH(100),F(1)
    26              REAL ICH,MINUS,IT
    27              DATA PLUS,ICH,MINUS,BLANK/1H+,1HI,1H-,1H /
    28       C
    29              I=1
    30              M1=M-1
    31              INCX1=INCX-1
    32       C
    33                 DO 100 J=1,M1
    34                 IT=ICH
    35                 IF((M-J)/INCY*INCY.EQ.M-J) IT=PLUS
    36                 GRAPH(1)=IT
    37       C
    38                    DO 20 JJ=2,N
    39                    GRAPH(JJ)=BLANK
    40       20           CONTINUE
    41       C
    42                 GRAPH(62)=1H.
    43       25        CONTINUE
    44                 IF(I.GT.IX) GO TO 50
    45                 IF(M-IFIX(.5+YS*(Y(I)-YMIN)).NE.J) GO TO 50
    46                 IXT=1+XS*(X(I)-XMIN)
    47                 GRAPH(IXT)=CHAR
    48                 I=I+1
    49                 GO TO 25
```

```
C
50      CONTINUE
        IF(IT.EQ.PLUS) GO TO 70
        WRITE(6,60) (GRAPH(JJ),JJ=1,N)
60      FORMAT(17X,100A1)
        GO TO 100
C
70      CONTINUE
        YT=(M-J)/YS+YMIN
        WRITE(6,80) YT,(GRAPH(JJ),JJ=1,N)
80      FORMAT(5X,E10.4,2X,100A1)
100     CONTINUE
C
        DO 120 J=1,N,INCX
C
           DO 110 JJ=1,INCX1
           JJJ=J+JJ
           GRAPH(JJJ)=MINUS
110        CONTINUE
C
        GRAPH(J)=PLUS
120     CONTINUE
C
125     CONTINUE
        IF(I.GT.IX) GO TO 150
        IF(M-IFIX(.5+YS*(Y(I)-YMIN)).NE.M) GO TO 150
        IXT=1+XS*(X(I)-XMIN)
        GRAPH(IXT)=CHAR
        I=I+1
        GO TO 125
C
150   CONTINUE
      WRITE(6,80) YMIN,(GRAPH(JJ),JJ=1,N)
      I=0
C
         DO 200 J=1,N,INCX
         I=I+1
         GRAPH(I)=XMIN+(J-1)/XS
200      CONTINUE
C
      WRITE(6,F) (GRAPH(J),J=1,I)
      RETURN
      END
```

```
1016*TEMP(1).GFILT
      1             SUBROUTINE GFILT(Y,N,L,W,Z)
      2       C
      3       C
      4       C          ROUTINE FOR A GAUSSIAN LOW PASS FILTER
      5       C
      6       C          Y.....INPUT FUNCTION
      7       C          N.....NUMBER OF INCREMENTS IN Y
      8       C          L.....LENGTH OF DOMAIN OF Y
      9       C          W.....WIDTH OF FILTER, STANDARD DEVIATION OF WINDOW
     10       C          Z.....FILTERED RESPONSE
     11       C
     12       C          NO EXTERNAL SUBROUTINES REQUIRED
     13       C
     14       C
     15             DIMENSION Y(1),Z(1),S(500)
     16       C
     17             AT=L/W/N/SQRT(6.28318)
     18             N1=N+1
     19             N2=15*W
     20             W2=2*W*W
     21       C
     22                DO 20 J=1,N2
     23                S(J)=AT*EXP(-J*J/W2)
     24       20       CONTINUE
     25       C
     26                DO 60 J=1,N
     27                ZT=Y(J)*AT
     28       C
     29                   DO 40 KK=1,N2
     30                   K=KK-(KK-1)/N*N
     31                   K1=J+K
     32                   K1=K1-(K1-1)/N*N
     33                   K2=N+J-K
     34                   K2=K2-(K2-1)/N*N
     35                   ZT=ZT+S(KK)*(Y(K1)+Y(K2))
     36       40          CONTINUE
     37       C
     38                Z(J)=ZT
     39       60       CONTINUE
     40       C
     41             RETURN
     42             END
```

```
28      1016*TEMP  PICTPR

      DIMENSION PICT(100,200),LABEL(2),THRESH(10)
     1,LINE(120),BND(600)
      INTEGER PICT
      DATA LABEL,IP/12HDATATAPE    ,100/
      EXTERNAL FMEAN
C
      WRITE(6,10)
10    FORMAT(' *** FILE NUMBER AND NAME ')
      READ(5,15) L,NAME
15    FORMAT(I2,1X,A6)
20    FORMAT()
C
      WRITE(6,30)
30    FORMAT(' *** FIRST AND LAST ROWS & COLUMNS')
      READ(5,20) N1,N2,M1,M2
      CALL INPIC(N1,N2,M1,M2,PICT,IP,LABEL,MIN,MAX,L)
C
      WRITE(6,40) MIN,MAX
40    FORMAT(' *** GRAY LEVEL RANGE ',2I4/' *** NUMBER OF',
     1' THRESHOLDS'/' *** WINDOW SIZE AND VERT. & HORIZ. SHIFT')
      READ(5,20) N,I1,I2,I3,I4
      WRITE(6,45)
45    FORMAT(' *** THRESHOLD VALUES')
      READ(5,20) (THRESH(I),I=1,N)
      WRITE(6,46)
46    FORMAT(' *** INVERT PICTURE')
      READ(5,48) INV
48    FORMAT(A1)
C
      N1=N2-N1+1
      M1=M2-M1+1
      CALL THRHLD(PICT,IP,I2,I1,M1,N1,I4,I3,THRESH,N,PICT,
     1IP,FMEAN,INV)
C
      CALL IMPRT(PICT,IP,N1,M1,N,LINE)
C
      WRITE(6,80)
80    FORMAT(' *** STARTING POINT AND SWEEP POSITION')
      READ(5,20) BND(1),BND(2),I1,I2
      CALL BNDTRC(PICT,BND,I1,I2,IP,ILIM)
      IL2=ILIM+ILIM
C
      WRITE(8,100) NAME,ILIM,(BND(J),J=1,IL2)
100   FORMAT(A6,1X,I3/(2F5.2))
      ENDFILE 8
C
      CALL SORT(BND,ILIM)
      WRITE(6,200) NAME
200   FORMAT(' *** EDGE CONTOUR FOR ',A6/)
      CALL PRNTBD(BND,ILIM,N1,M1,LINE)
```

```
                28      1016*TEMP  PICTPR

51          C
52                  STOP
53                  END
```

```
      20     1016*MISC  INPIC

 1            SUBROUTINE INPIC(N1,N2,M1,M2,PICT,IP,LABEL,MIN,MAX,NF)
 2      C
 3      C
 4      C     THIS ROUTINE READS FROM TAPE A PICTURE SEGMENT.
 5      C
 6      C     PARAMETERS:
 7      C        N1.....FIRST ROW
 8      C        N2.....LAST ROW
 9      C        M1.....FIRST COLUMN
10      C        M2.....LAST COLUMN
11      C        PICT....ARRAY RETURNING PICTURE SEGMENT
12      C        IP.....NUMBER OF ROWS IN PICT
13      C        LABEL...TWO ELEMENT ARRAY CONTAINING FILE NAME (LEFT JSTFD)
14      C        MIN.....MINIMUM GRAY SCALE VALUE IN SEGMENT
15      C        MAX.....MAXIMUM GRAY SCALE VALUE IN SEGMENT
16      C        NF......FILE NUMBER
17      C
18      C     SUBROUTINES REQUIRED: FLDTAP,UNPACK
19      C
20      C
21            DIMENSION PICT(IP,1),LABEL(1),IN(114),OUT(512)
22            INTEGER PICT,OUT
23      C
24            MAX=0
25            MIN=300
26      C
27            CALL FLDTAP(11,LABEL)
28            CALL FLDTAP(6)
29            CALL FLDTAP(5)
30            CALL FLDTAP(1)
31      C
32            L=N1+NF*241
33            CALL FLDTAP(9,L)
34            ND=N2+1-N1
35      C
36               DO 50 J=1,ND
37               CALL FLDTAP(7,114,IN,OUT)
38               CALL UNPACK(IN,OUT,M2,8)
39               I=0
40      C
41                  DO 30 L=M1,M2
42                  I=I+1
43                  MT=OUT(L)
44                  IF(MT.LT.MIN) MIN=MT
45                  IF(MT.GT.MAX) MAX=MT
46                  PICT(J,I)=MT
47      30          CONTINUE
48      C
49      50       CONTINUE
50      C
```

```
20      1016*MISC  IN-IC

51              CALL FLETAP(2)
52              RETURN
53              END
```

```
51      1016*MISC  UNPACK

 1              SUBROUTINE UNPACK(IN,OUT,IW,IB)
 2              DIMENSION IN(1),OUT(1)
 3              INTEGER OUT
 4              I=1
 5              I1=0
 6      C
 7                 DO 100 J=1,IW
 8                 IF(I1+IB.GT.36) GO TO 50
 9                 OUT(J)=FLD(I1,IB,IN(I))
10                 I1=I1+IB
11                 IF(I1.LE.35) GO TO 100
12                 I=I+1
13                 I1=0
14                 GO TO 100
15      50         CONTINUE
16                 IBT=36-I1
17                 OT=FLD(I1,IBT,IN(I))
18                 I=I+1
19                 IBT=IB-IBT
20                 I1=0
21                 OUT(J)=2**IBT*OT+FLD(I1,IBT,IN(I))
22                 I1=I1+IBT
23      100        CONTINUE
24      C
25              RETURN
26              END
```

```
                48      1016*MISC  THRHLD

 1                SUBROUTINE THRHLD(PICT,IP,XW,YW,XM,YM,HS,VS,THRESH,
 2               1NT,TPICT,IT,F,INV)
 3       C
 4       C
 5       C       ROUTINE TO THRESHOLD A PICTURE SEGMENT
 6       C
 7       C       PICT....INPUT PICTURE ARRAY
 8       C       IP......NUMBER OF ROWS PER COLUMN IN PICT
 9       C       XW......HORIZONTAL WINDOW WIDTH
10       C       YW......VERTICAL WINDOW HEIGHT
11       C       XM......NUMBER OF COLUMNS IN PICTURE SEGMENT
12       C       YM......NUMBER OF ROWS IN PICTURE SEGMENT
13       C       HS......HORIZONTAL INCREMENTAL SHIFT OF WINDOW
14       C       VS......VERTICAL INCREMENTAL SHIFT OF WINDOW
15       C       THRESH..ARRAY OF THRESHOLD VALUES
16       C       NT......NUMBER OF THRESHOLD VALUES
17       C       TPICT...THRESHOLDED PICTURE ARRAY
18       C       IT......NUMBER OF ROWS IN TPICT
19       C       F.......WINDOW FUNCTION TO BE THRESHOLDED
20       C       INV.....YES IF PICTURE LEVELS ARE TO BE INVERTED
21       C
22       C       REQUIRES SUBROUTINES NOTHR & F (EXTERNAL)
23       C
24       C
25                DIMENSION PICT(1),TPICT(1),THRESH(1)
26                INTEGER XW,YW,XM,YM,HS,VS,PICT,TPICT
27       C
28                IXM=XM-XW+1
29                IYM=YM-YW+1
30                IX=-IT
31                KX=0
32                KXS=HS*IP
33       C
34                   DO 100 J1=1,IXM,HS
35                   KXY=KX+1
36                   IX=IX+IT
37                   IXY=IX
38       C
39                      DO 50 J2=1,IYM,VS
40                      IXY=IXY+1
41                      TH=F(PICT(KXY),IP,XW,YW)
42                      TPICT(IXY)=NOTHR(THRESH,NT,TH,INV)
43                      KXY=KXY+VS
44       50             CONTINUE
45       C
46                   KX=KX+KXS
47       100         CONTINUE
48       C
49                XM=(IXM-1)/HS+1
50                YM=(IYM-1)/VS+1
```

```
48      1016*MISC  THRHLD

51              RETURN
52              END
```

```
      FUNCTION NOTHR(T,IT,TH,INV)
C
C
C     ROUTINE TO THRESHOLD TH
C
C     T.......ARRAY OF THRESHOLD VALUES
C     IT......NUMBER OF THRESHOLD VALUES
C     TH......VALUE TO BE THRESHOLDED
C     NOTHR...NUMBER OF SEGMENT CONTAINING TH. (0,1,...,IT)
C     INV.....YES IF PICTURE LEVELS ARE TO BE INVERTED
C
C     NO EXTERNAL SUBROUTINES REQUIRED
C
C
      DIMENSION T(1)
C
         DO 50 J=1,IT
         NOTHR=J
         IF(T(J).GT.TH) GO TO 100
50       CONTINUE
C
         GO TO 150
100   CONTINUE
      NOTHR=NOTHR-1
150   CONTINUE
      IF(INV.EQ.1HY) NOTHR=IT-NOTHR
      RETURN
      END
```

```
                18      1016*MISC  IMPRT

 1                      SUBROUTINE IMPRT(PICT,IP,N1,M1,N,LINE)
 2          C
 3          C
 4          C     ROUTINE TO PRINT PICTURE ON LINE PRINTER
 5          C
 6          C     PICT....PICTURE ARRAY
 7          C     IP......NUMBER OF ROWS IN PICT
 8          C     N1......NUMBER OF LINES IN PICTURE
 9          C     M1......NUMBER OF CHARACTERS PER LINE
10          C     N.......NUMBER OF THRESHOLD VALUES
11          C     LINE....SCRATCH ARRAY WITH AT LEAST 120 ELEMENTS
12          C
13          C     NO EXTERNAL SUBROUTINES REQUIRED
14          C
15          C
16                      DIMENSION PICT(IP,1),LINE(1),ICHAR(10)
17                      INTEGER PICT
18                      DATA ICHAR/1H ,1H',1H!,1H%,1HO,1H+,1H*,1HX,1H#,1H$/
19          C
20                      A=9.0/N
21          C
22                      DO 100 J=1,N1
23          C
24                         DO 50 I=1,M1
25                         K=PICT(J,I)*A+1
26                         LINE(I)=ICHAR(K)
27          50             CONTINUE
28          C
29                      WRITE(6,60) (LINE(I),I=1,M1)
30          60      FORMAT(5X,120A1)
31          100     CONTINUE
32          C
33                      RETURN
34                      END
```

```
                2      1016*MISC  BNDTRC

 1              SUBROUTINE BNDTRC(PICT,BND,IX,IY,N,LIM)
 2      C
 3      C
 4      C      ROUTINE TO TRACE IMAGE BOUNDARY
 5      C
 6      C      PICT....(0,1) BINARY PICTURE, REPRESENTING IMAGE
 7      C      BND.....RETURNED IMAGE POINTS
 8      C      BND(1)..STARTING X COORDINATE, SUPPLIED BY CALLING
 9      C                ROUTINE, NUMBER OF COLUMNS TO LEFT OF RIGHT
10      C                EDGE OF PICTURE
11      C      BND(2)..STARTING Y COORDINATE, NUMBER OF LINES FROM
12      C                TOP OF PICTURE
13      C      IX......STARTING POSITION OF SWEEP RAY, RELATIVE TO
14      C                TO STARTING X COORDINATE
15      C      IY......STARTING POSITION OF SWEEP RAY, RELATIVE TO
16      C                TO STARTING Y COORDINATE
17      C      N.......NUMBER OF ROWS IN PICTURE
18      C      LIM.....NUMBER OF BOUNDARY POINTS FOUND
19      C
20      C      NO EXTERNAL SUBROUTINES REQUIRED
21      C
22      C
23             DIMENSION PICT(N,1),BND(1)
24             INTEGER X1,Y1,X,Y,PICT,XT,YT
25      C
26             X1=BND(1)
27             Y1=BND(2)
28             X=X1
29             Y=Y1
30             I=3
31      C
32      10        CONTINUE
33                IXL=IX
34                IYL=IY
35                IF(IX*IY.LE.0.AND.IY.NE.0) GO TO 20
36                IY=IY-IX
37                GO TO 30
38      20        CONTINUE
39                IX=IY+IX
40      30        CONTINUE
41      C
42                XT=X+IX
43                YT=Y-IY
44                IF(XT.LE.0.OR.YT.LE.0) GO TO 10
45                IF(PICT(YT,XT)) 10,10,40
46      40        CONTINUE
47                IF(X1.EQ.XT.AND.YT.EQ.Y1) GO TO 50
48                BND(I)=XT
49                I=I+1
50                BND(I)=YT
```

```
51                  I=I+1
52                  X=XT
53                  Y=YT
54                  IX=IXL-IX
55                  IY=IYL-IY
56                  GO TO 10
57          C
58          50      CONTINUE
59                  LIM=(I-1)/2
60                  RETURN
61                  END
```

```
      44      1016*MISC  SORT

 1              SUBROUTINE SORT(BOUND,ILIM)
 2        C
 3        C
 4        C     ROUTINE TO SORT BOUNDARY POINTS SUCH THAT Y VALUES
 5        C     (EVEN INDICES) ARE IN DECREASING ORDER
 6        C
 7        C     BOUND...ARRAY OF BOUNDARY POINTS (TWO ELEMENTS/POINT)
 8        C     ILIM....NUMBER OF BOUNDARY POINTS
 9        C
10        C     NO EXTERNAL SUBROUTINES REQUIRED
11        C
12        C
13              DIMENSION BOUND(1)
14        C
15              ILIM1=ILIM-1
16        C
17                 DO 100 J=1,ILIM1
18                 JJ=J+J
19                 JJ1=JJ-1
20                 J1=J+1
21                 B1=BOUND(JJ)
22        C
23                    DO 80 K=J1,ILIM
24                    KK=K+K
25                    B2=BOUND(KK)
26                    IF(B1-B2) 80,80,40
27        40          CONTINUE
28                    BOUND(KK)=B1
29                    B1=B2
30                    BOUND(JJ)=B2
31                    B2=BOUND(JJ1)
32                    KK1=KK-1
33                    BOUND(JJ1)=BOUND(KK1)
34                    BOUND(KK1)=B2
35        80          CONTINUE
36        C
37        100      CONTINUE
38        C
39              RETURN
40              END
```

```
31      1016*MISC  PRNTBD

      1             SUBROUTINE PRNTBD(BND,LIM,N,M,LINE)
      2       C
      3       C
      4       C     ROUTINE TO PRINT BOUNDARY CONTOUR
      5       C
      6       C     BND.....ARRAY OF BOUNDARY POINTS
      7       C     LIM.....NUMBER OF BOUNDARY POINTS
      8       C     N.......NUMBER OF LINES IN PICTURE
      9       C     M.......NUMBER OF CHARACTERS PER LINE
     10       C     LINE....WORKING ARRAY OF NOT LESS THAN M ELEMENTS
     11       C
     12       C     NO EXTERNAL SUBROUTINES REQUIRED
     13       C
     14       C
     15             DIMENSION BND(1),LINE(1)
     16             INTEGER STAR,BLNK
     17             DATA STAR/1H*/,BLNK/1H /
     18       C
     19             LL=LIM+LIM
     20             I=2
     21       C
     22                DO 100 J=1,N
     23       C
     24                   DO 20 K=1,M
     25                   LINE(K)=BLNK
     26       20          CONTINUE
     27       C
     28       30       CONTINUE
     29                IF(I.GT.LL) GO TO 70
     30                L1=BND(I)
     31                IF(L1.NE.J) GO TO 70
     32                I1=I-1
     33                L1=BND(I1)
     34                LINE(L1)=STAR
     35                I=I+2
     36                GO TO 30
     37       70       CONTINUE
     38                WRITE(6,80) (LINE(K),K=1,M)
     39       80       FORMAT(5X,100A1)
     40       100      CONTINUE
     41       C
     42             RETURN
     43             END
```

```
1016*MISC(1),COPY
      1              DIMENSION BUF(300),K(6)
      2              WRITE(6,2)
      3        2     FORMAT(' *** NUMBER OF RECORDS PER FILE AND'
      4             1,' NUMBER OF WORDS PER RECORD')
      5              READ(5,20) NREC,NW
      6              NNN=NREC+2
      7              CALL FLDTAP(11,12HSOURCE      )
      8              CALL FLDTAP(6)
      9              CALL FLDTAP(5)
     10              CALL FLDTAP(1)
     11              CALL FLDTAP(9,NNN)
     12        C
     13              WRITE(6,10)
     14        10    FORMAT(' *** NUMBER OF FILES TO COPY')
     15              READ(5,20) N
     16        20    FORMAT()
     17              WRITE(6,12)
     18        12    FORMAT(' *** REWIND OUTPUT FILE?')
     19              READ(5,14) ANS
     20        14    FORMAT(A1)
     21              IF(ANS.EQ.1HY) CALL NTRAN(10,10)
     22        C
     23                 DO 100 J=1,N
     24        C
     25                    DO 50 I=1,NREC
     26                    CALL FLDTAP(7,NW,BUF,K)
     27                    CALL NTRAN(10,1,NW,BUF,L)
     28        50          CONTINUE
     29        C
     30                 CALL FLDTAP(9,1)
     31                 CALL NTRAN(10,9)
     32        100      CONTINUE
     33        C
     34              CALL FLDTAP(2)
     35              STOP
     36              END
-END PRT
```

SECTION 9

SAMPLE TRACKING IMAGERY

Twenty video images have been digitized at 512 x 240 pixels, 8-bits per pixel, and are stored on magnetic tape and on floppy disks at WSMR. These and future digitized images to be taken from sequential video frames can be made available upon request. The image names are as follows:

MI1V1 - Missile 1, View 1 (Before take-off)
MI1V2 - Missile 1, View 2
MI1V3 - Missile 1, View 3
MILV4 - Missile 1, View 4
MI2V1 - Missile 2, View 1 (Before take-off)
MI2V2 - Missile 2, View 2
MI2V3 - Missile 2, View 3 (Field 1)
MI2V4 - Missile 2, View 4 (Field 2 of MI2V3)
MI3V1 - Missile 3, View 1
MI3V2 - Missile 3, View 2
PL1V1 - Plane 1, View 1
PL1V2 - Plane 1, View 2
PL1V3 - Plane 1, View 3
PL2V1 - Plane 2, View 1
PL2V2 - Plane 2, View 2
PL2V3 - Plane 2, View 3
PL2V4 - Plane 2, View 4
CR1V1 - Cruise Missile 1, View 1
CR1V2 - Cruise Missile 1, View 2
CR1V3 - Cruise Missile 1, View 3

Figures 9.1 through 9.20 are gray level half-tone copies of these 20 images produced on the Tektronix 4014 display terminal using WRP512, a 17 gray level display program. A gray level histogram of each digitized image is also included as part b of each figure.

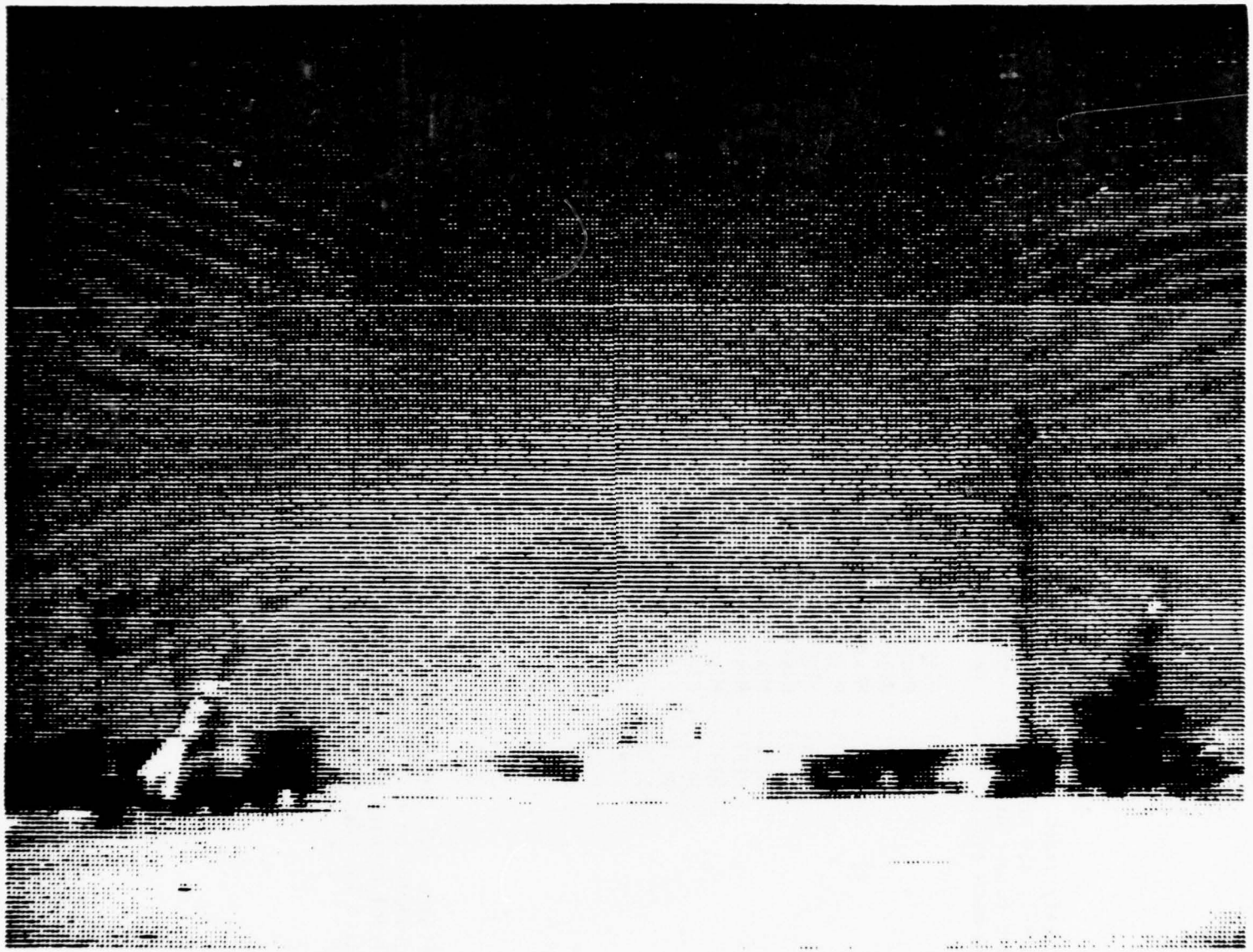


Figure 9.1a. Grey Scale Display of Missile 1, View 1

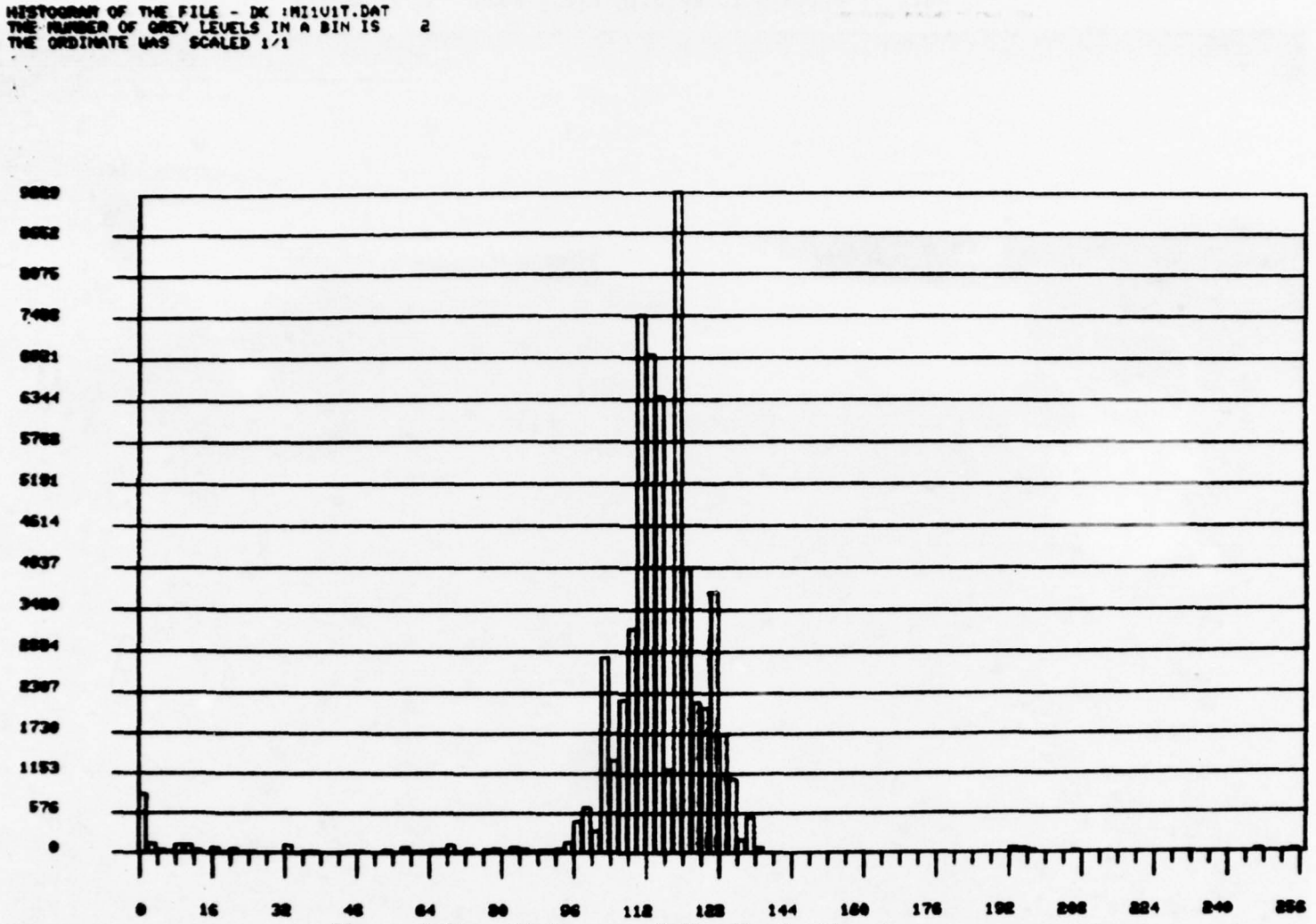


Figure 9.1b. Histogram of Missile 1, View 1

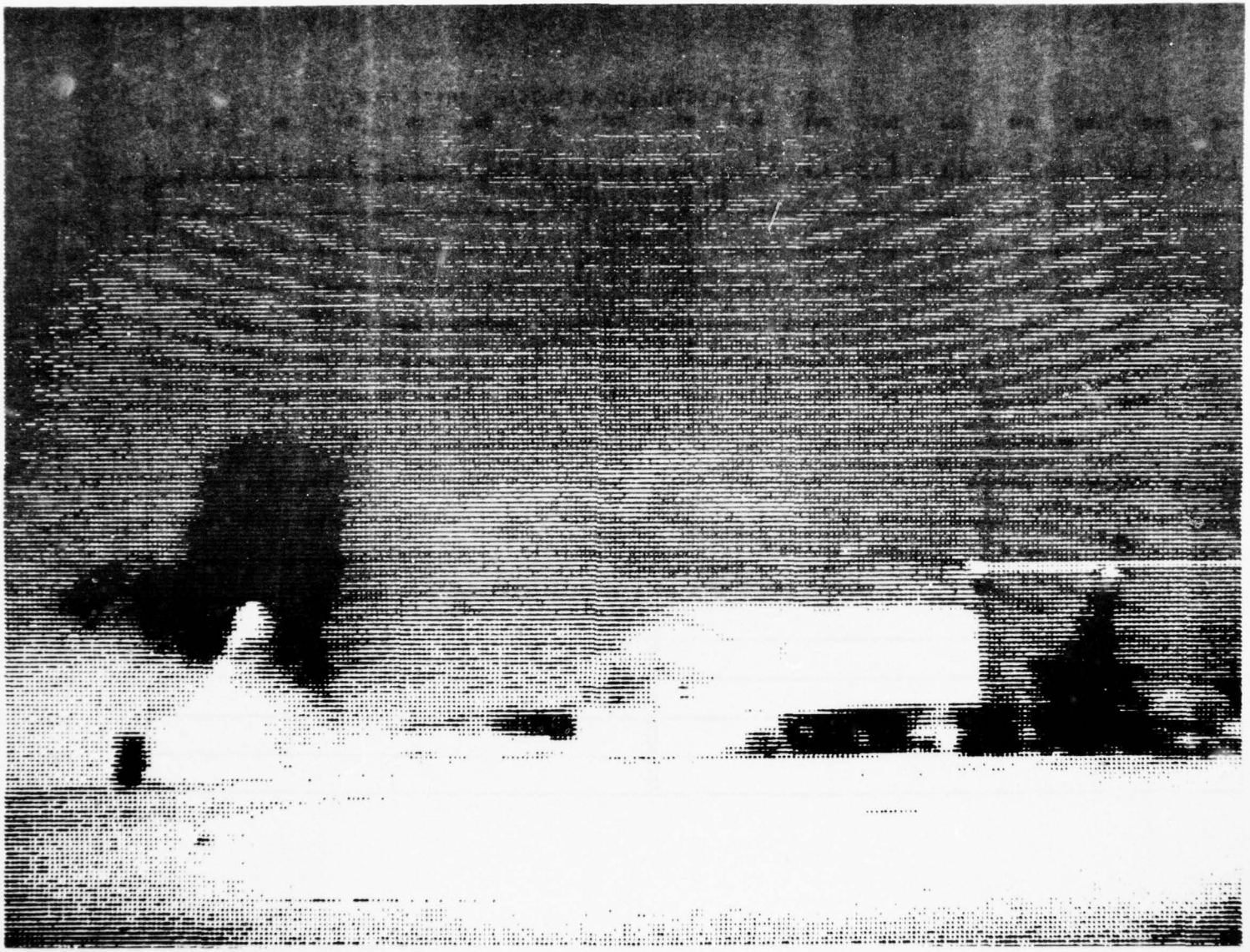


Figure 9.2a. Grey Scale Display of Missile 1, View 2

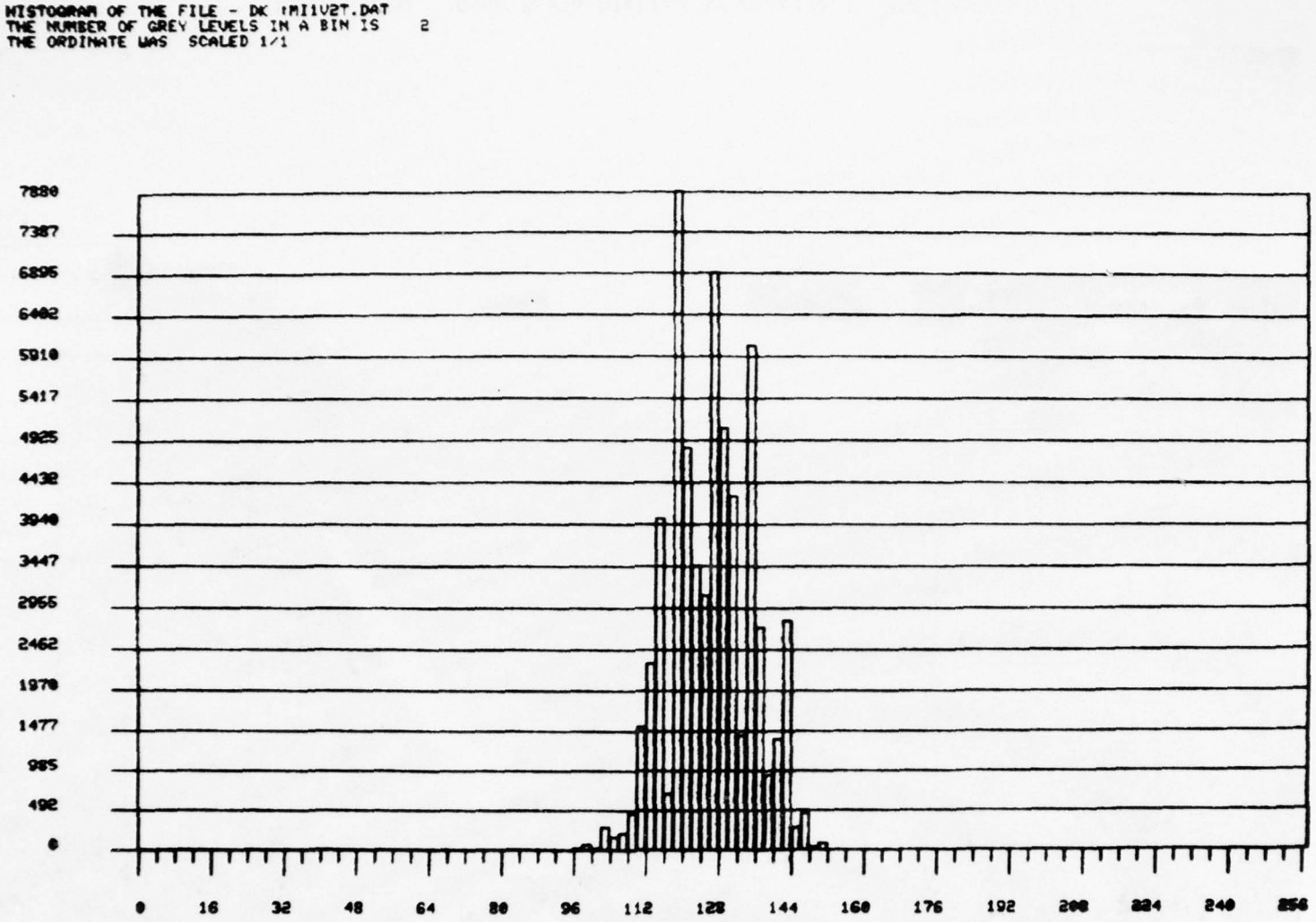


Figure 9.2b. Histogram of Missile 1, View 2

Figure 9.3a. Grey Scale Display of Missile 1, View 3

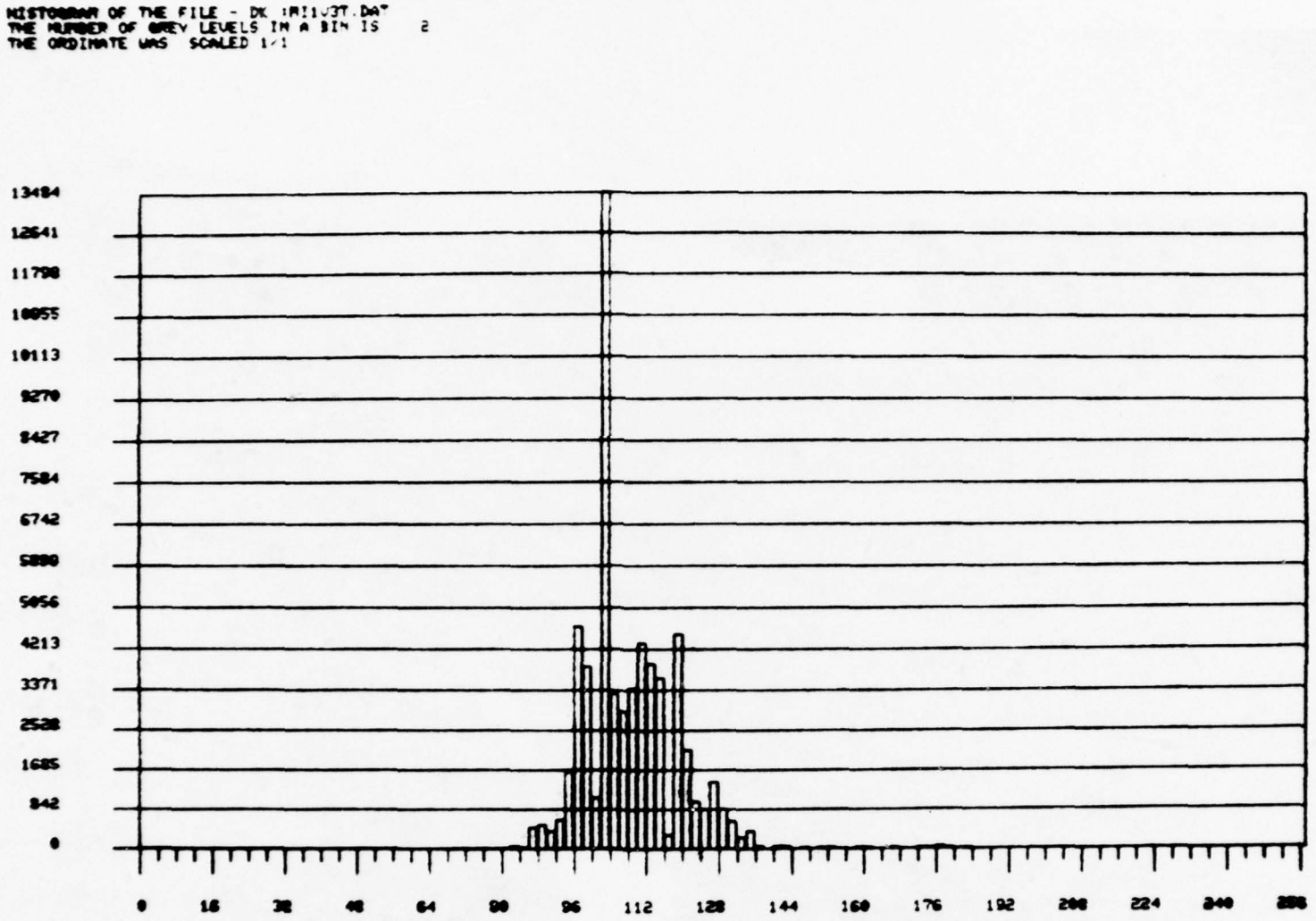


Figure 9.3b. Histogram of Missile 1, View 3

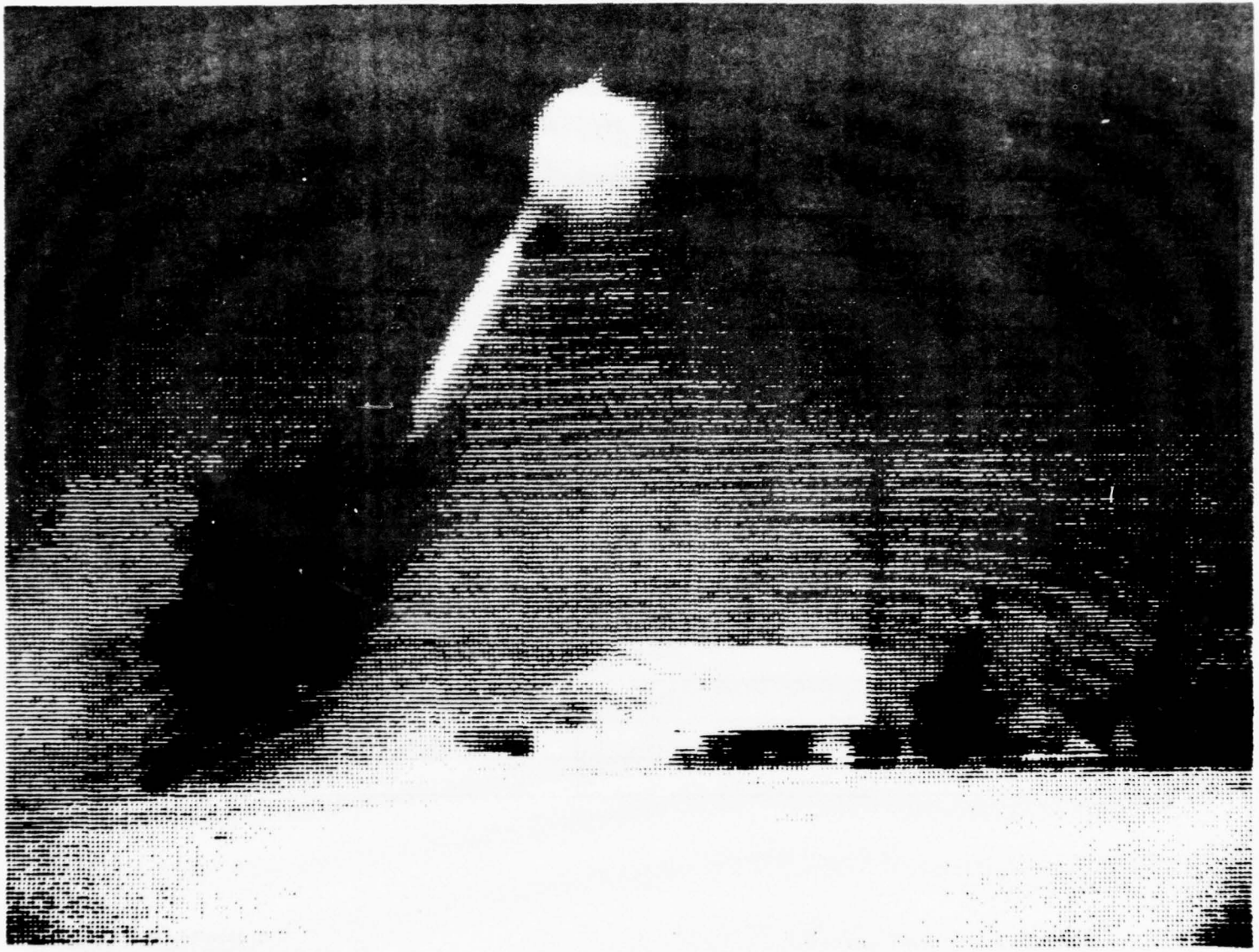


Figure 9.4a. Grey Scale Display of Missile 1, View 4

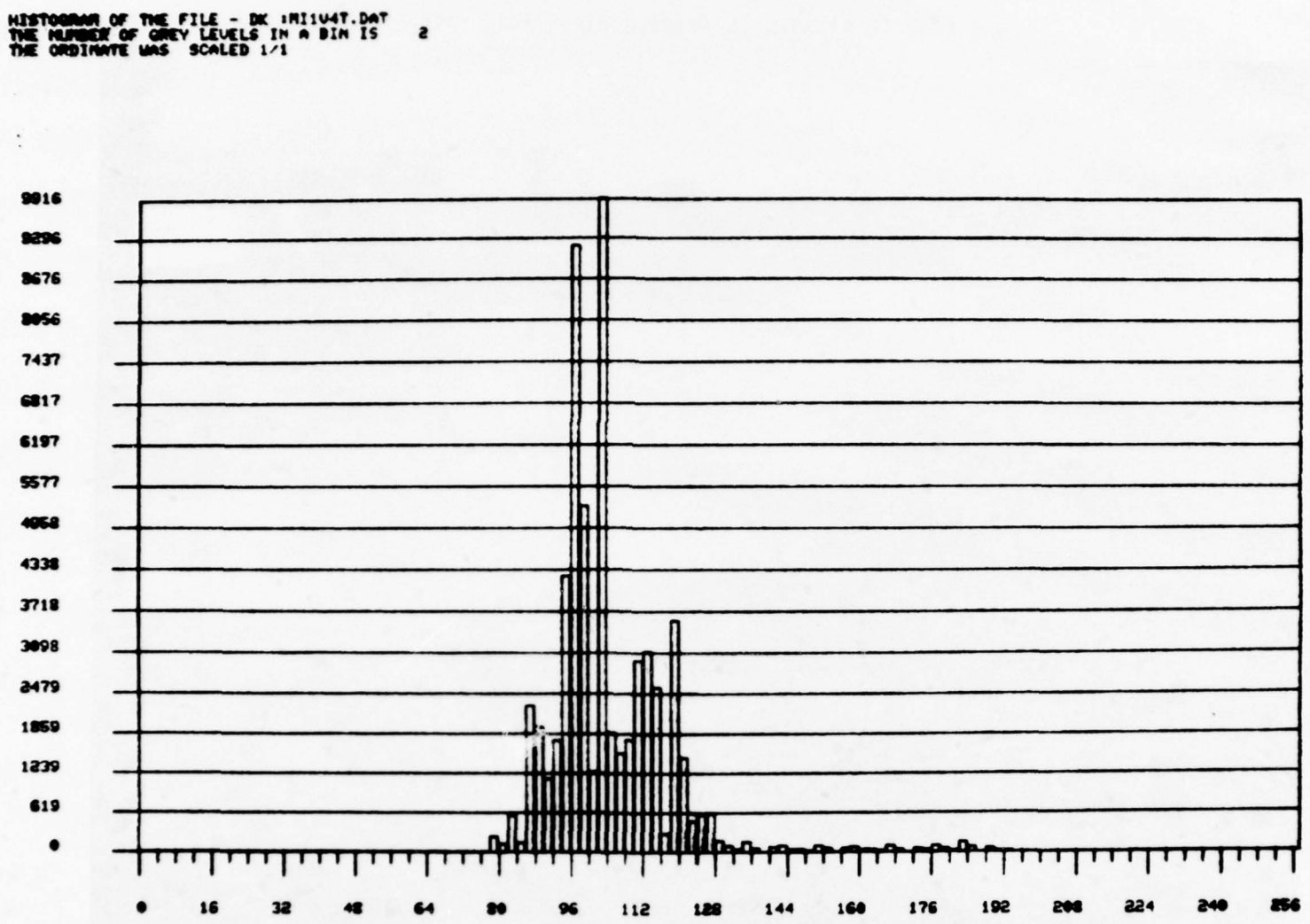


Figure 9.4b. Histogram of Missile 1, View 4

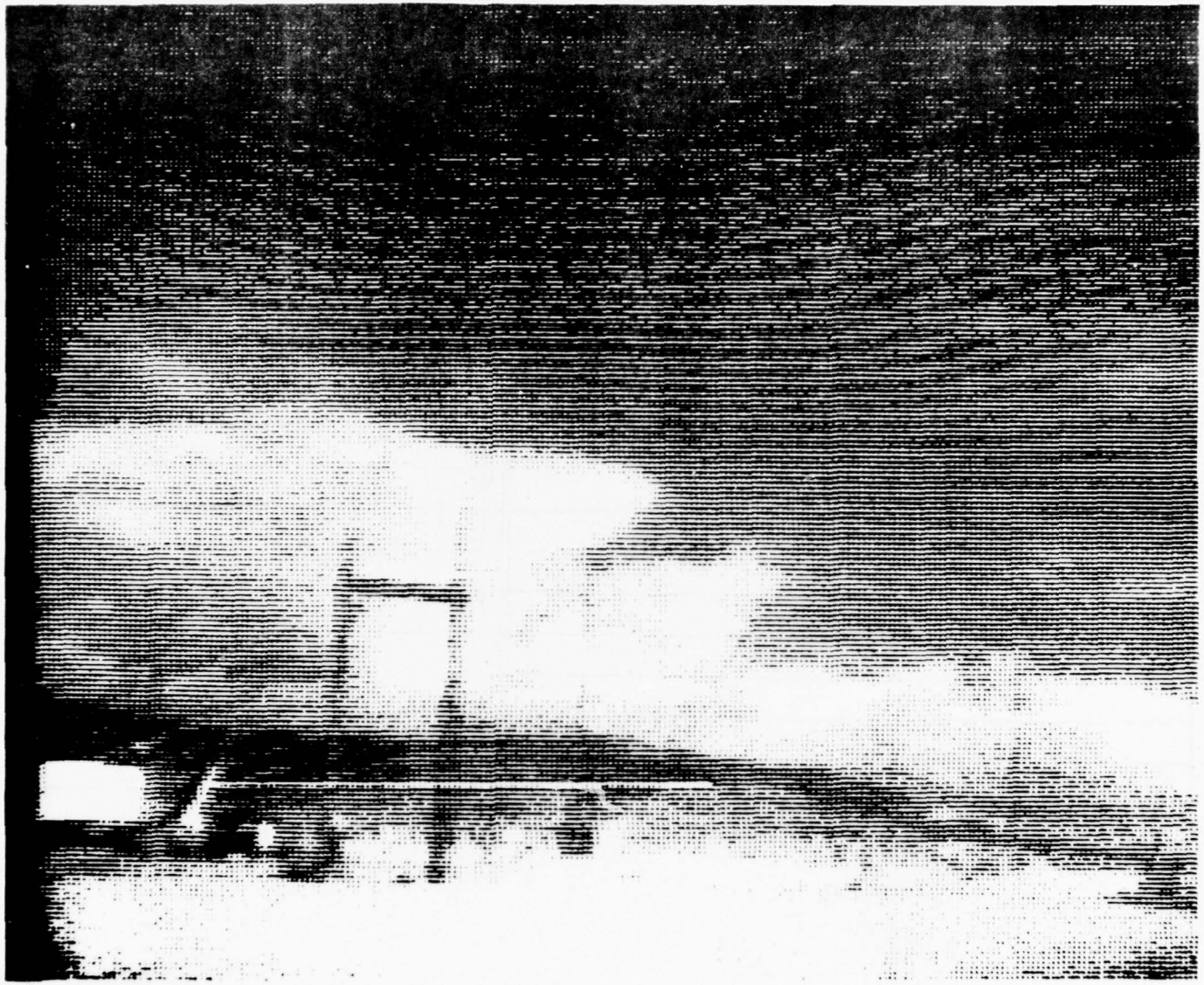


Figure 9.5a. Grey Scale Display of Missile 2, View 1

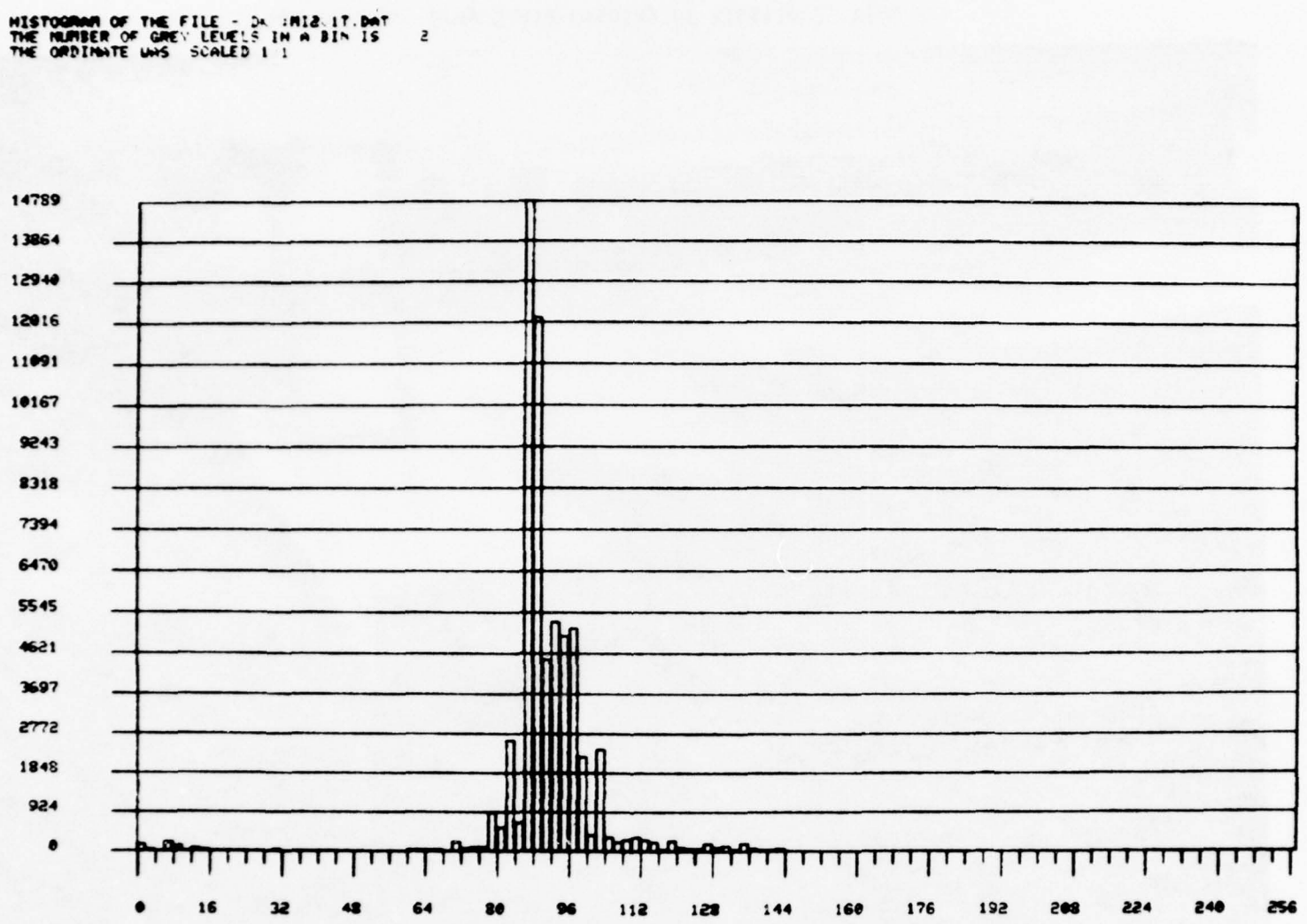


Figure 9.5b. Histogram of Missile 2, View 1

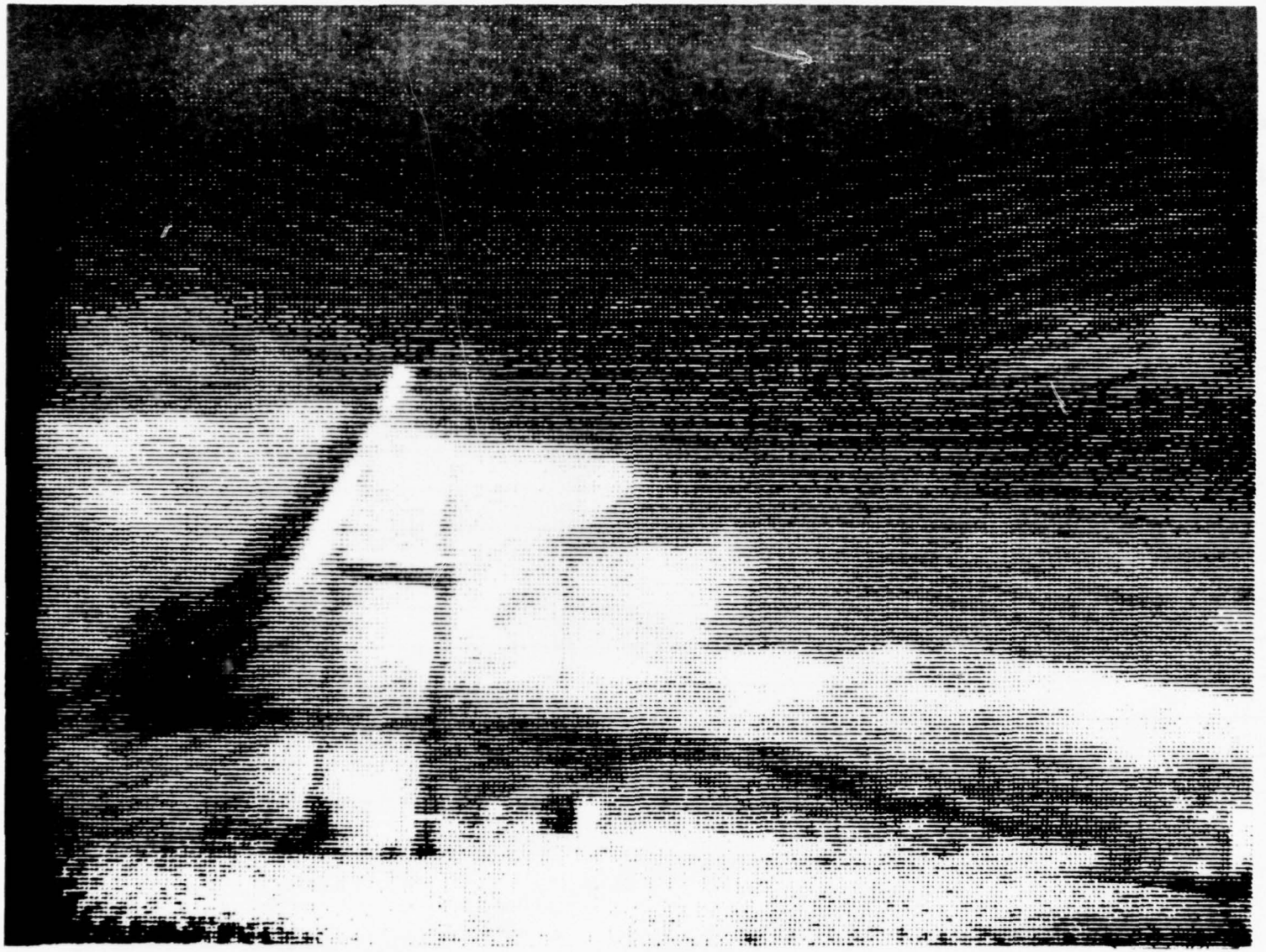


Figure 9.6a. Grey Scale Display of Missile 2, View 2

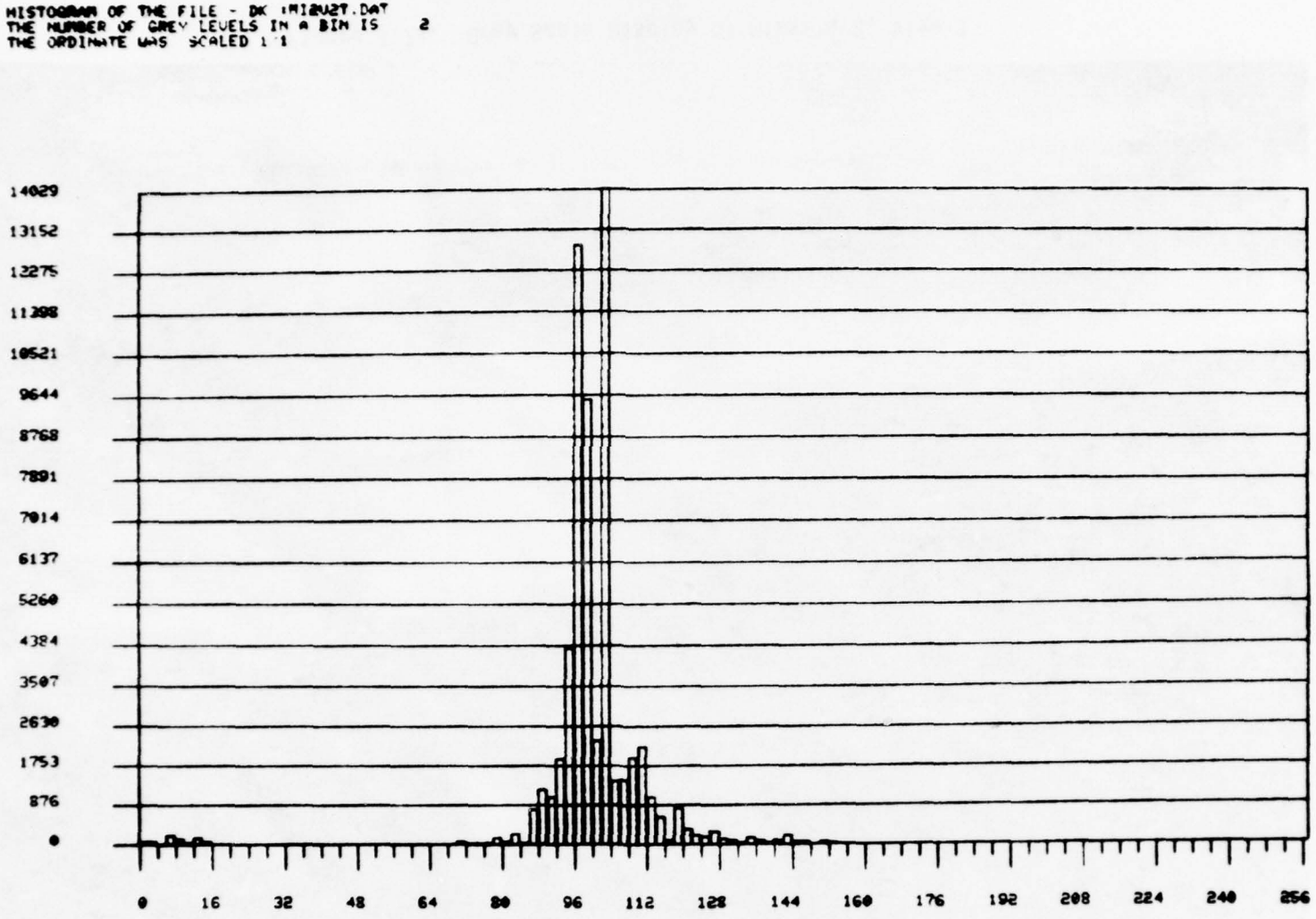


Figure 9.6b. Histogram of Missile 2, View 2

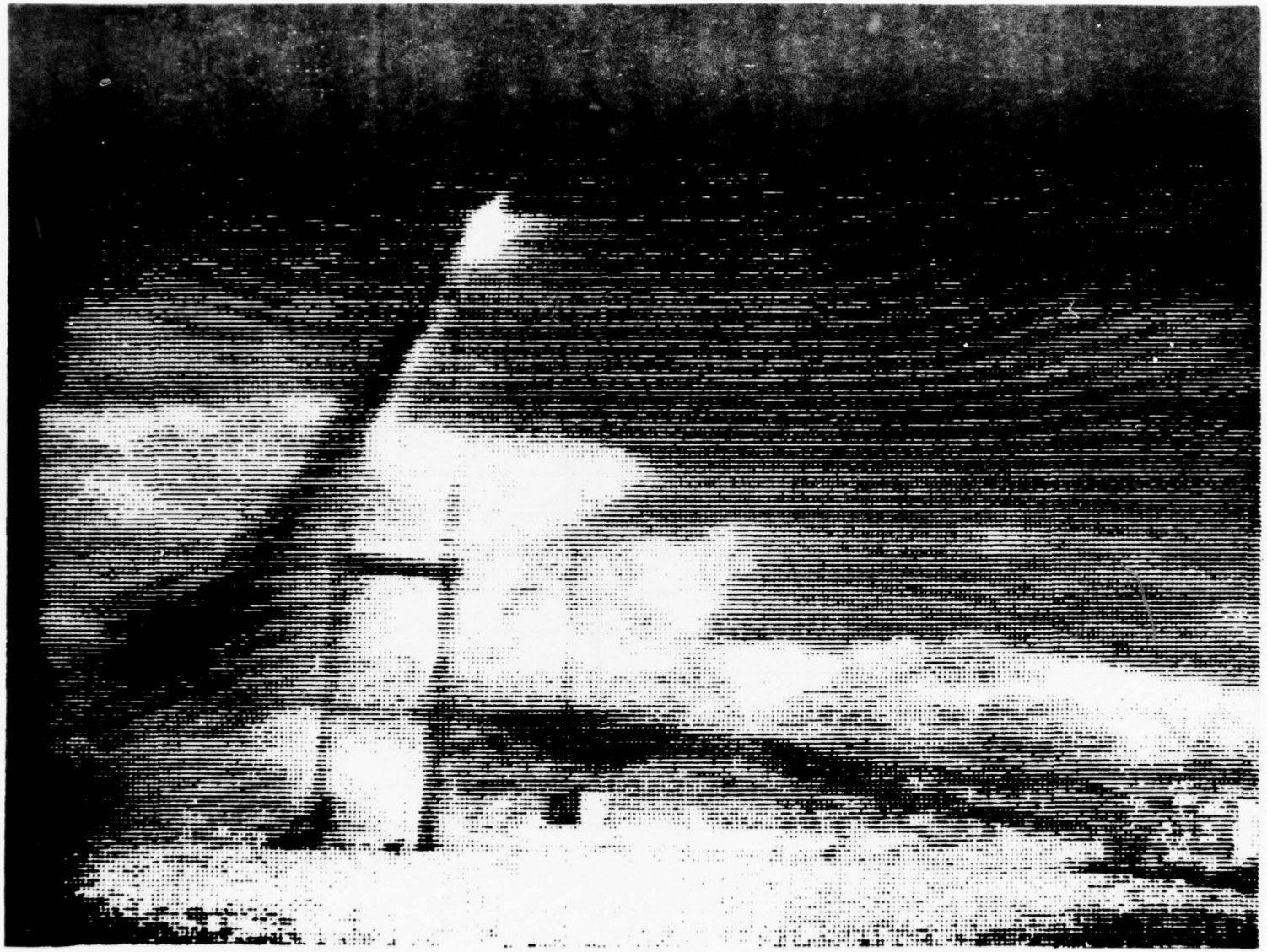


Figure 9.7a. Grey Scale Display of Missile 2, View 3

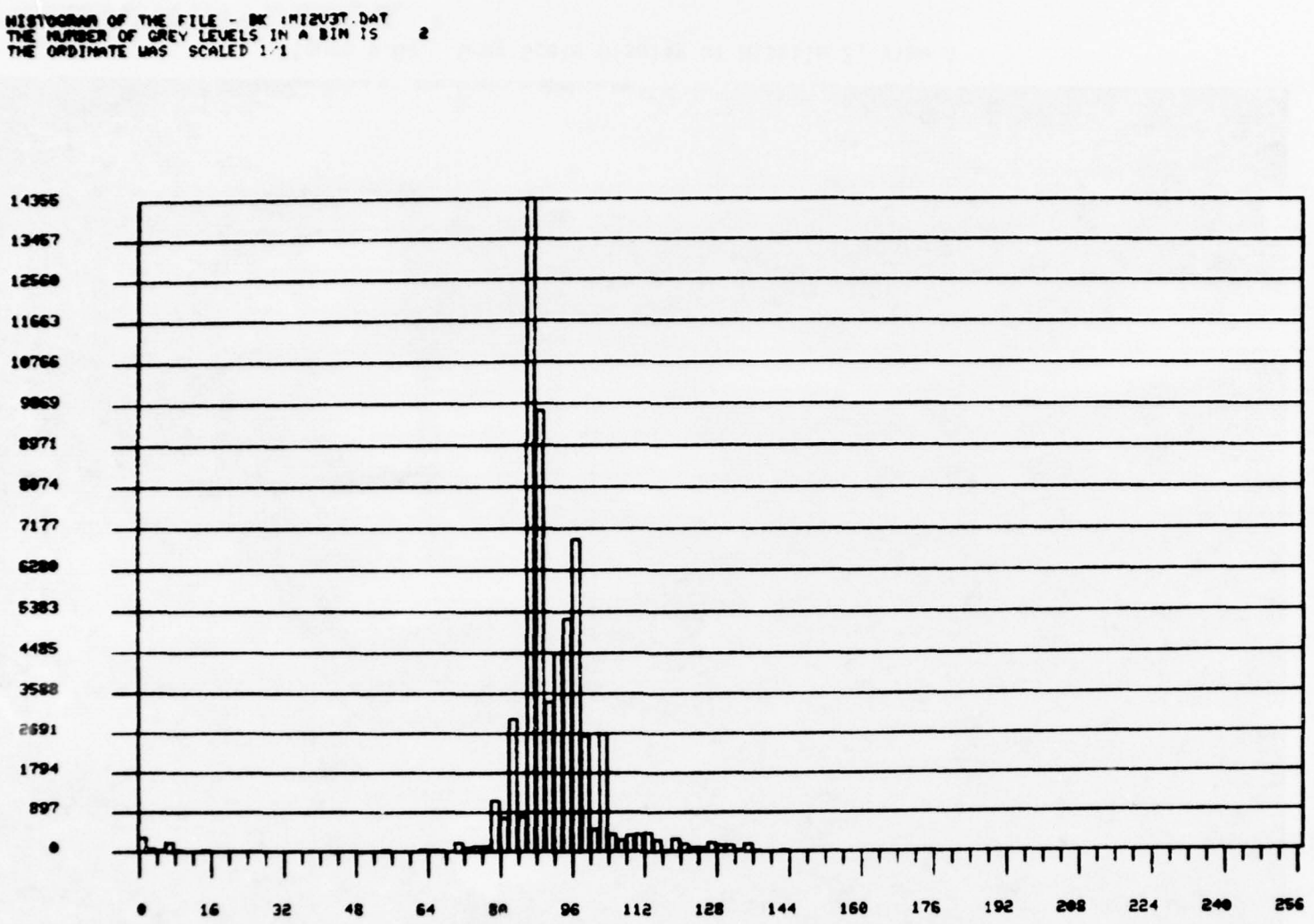


Figure 9.7b. Histogram of Missile 2, View 3

Figure 9.8a. Grey Scale Display of Missile 2, View 4

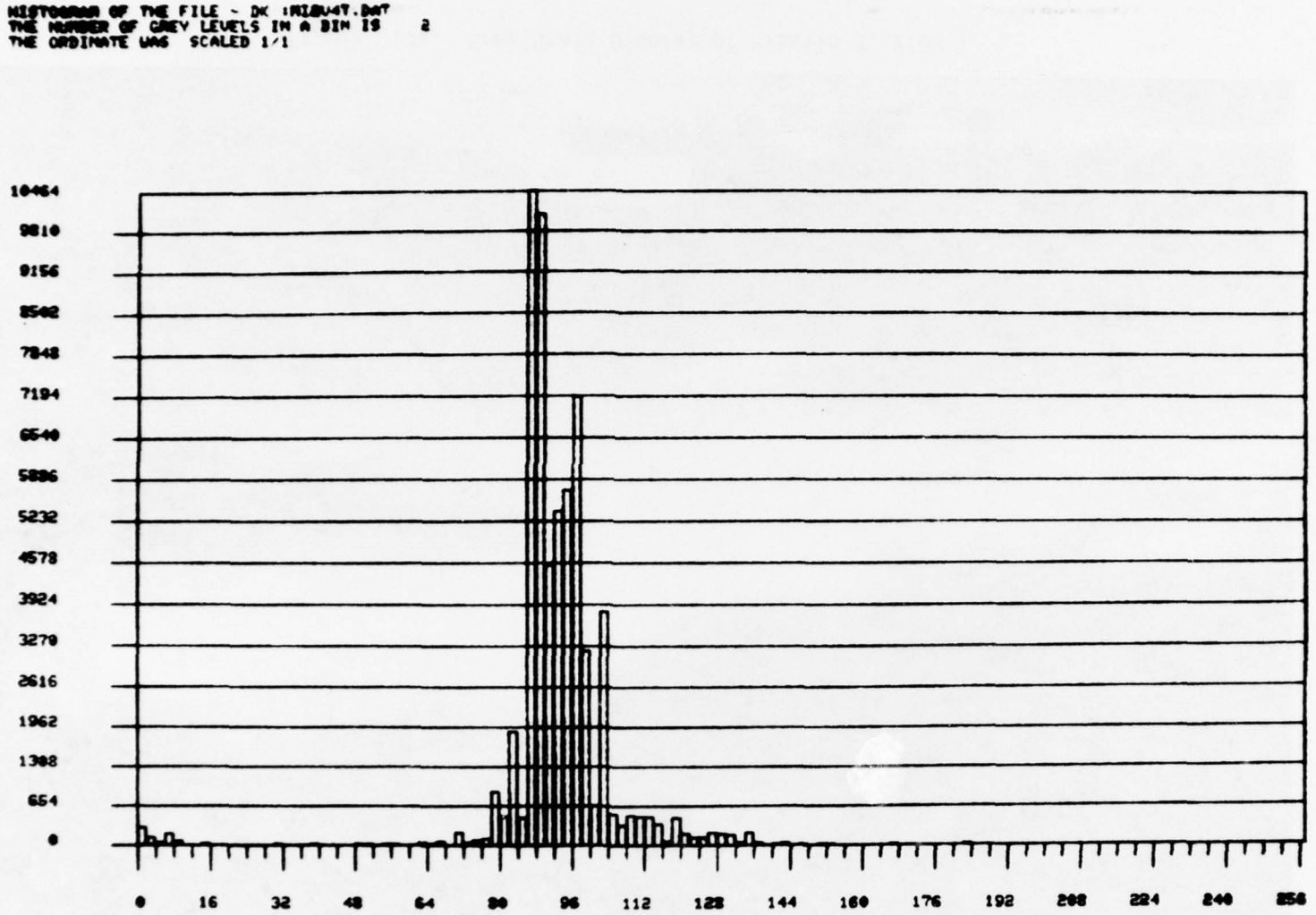


Figure 9.8b. Histogram of Missile 2, View 4

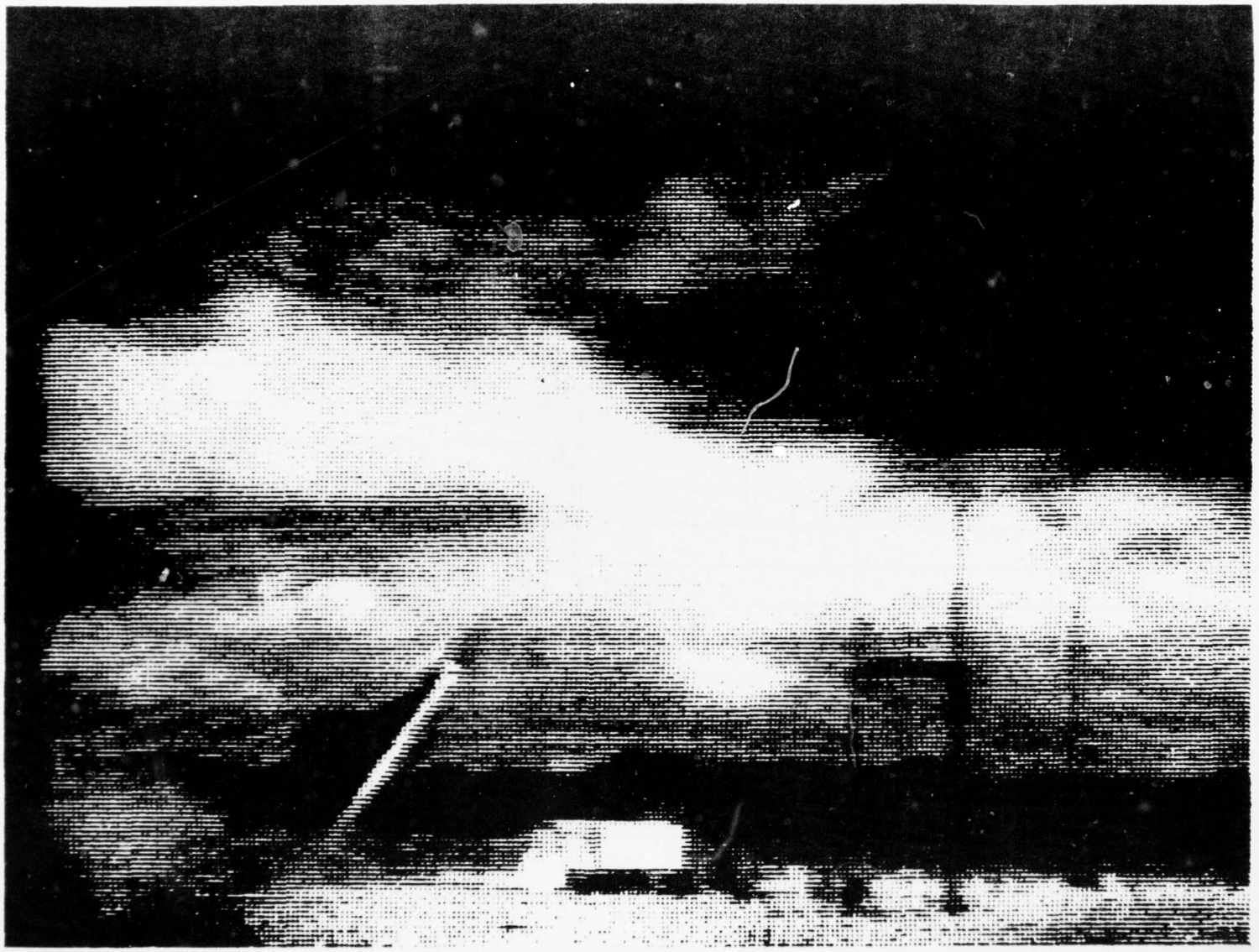


Figure 9.9a. Grey Scale Display of Missile 3, View 1

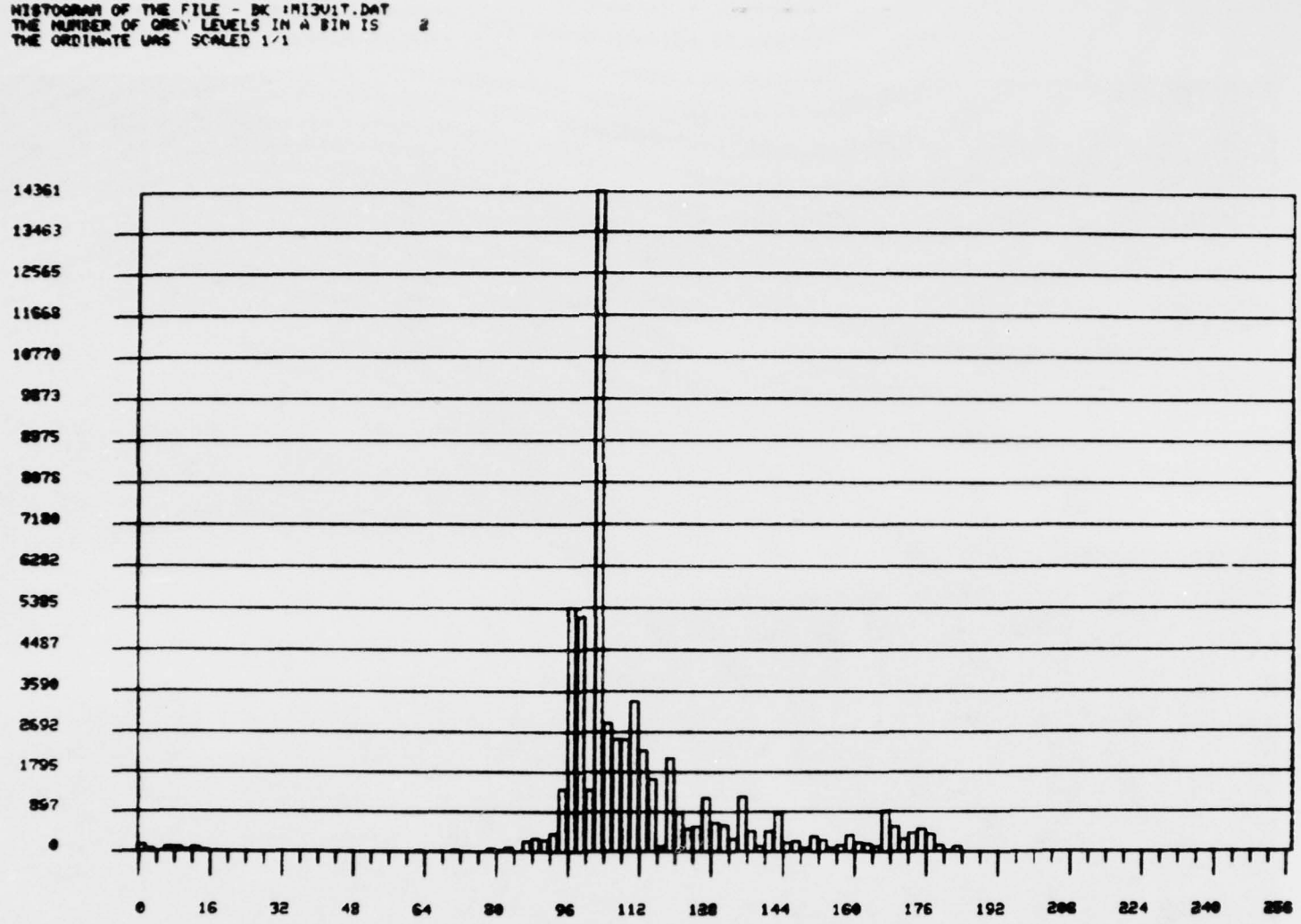


Figure 9.9b. Histogram of Missile 3, View 1

Figure 9.10a. Grey Scale Display of Missile 3, View 2

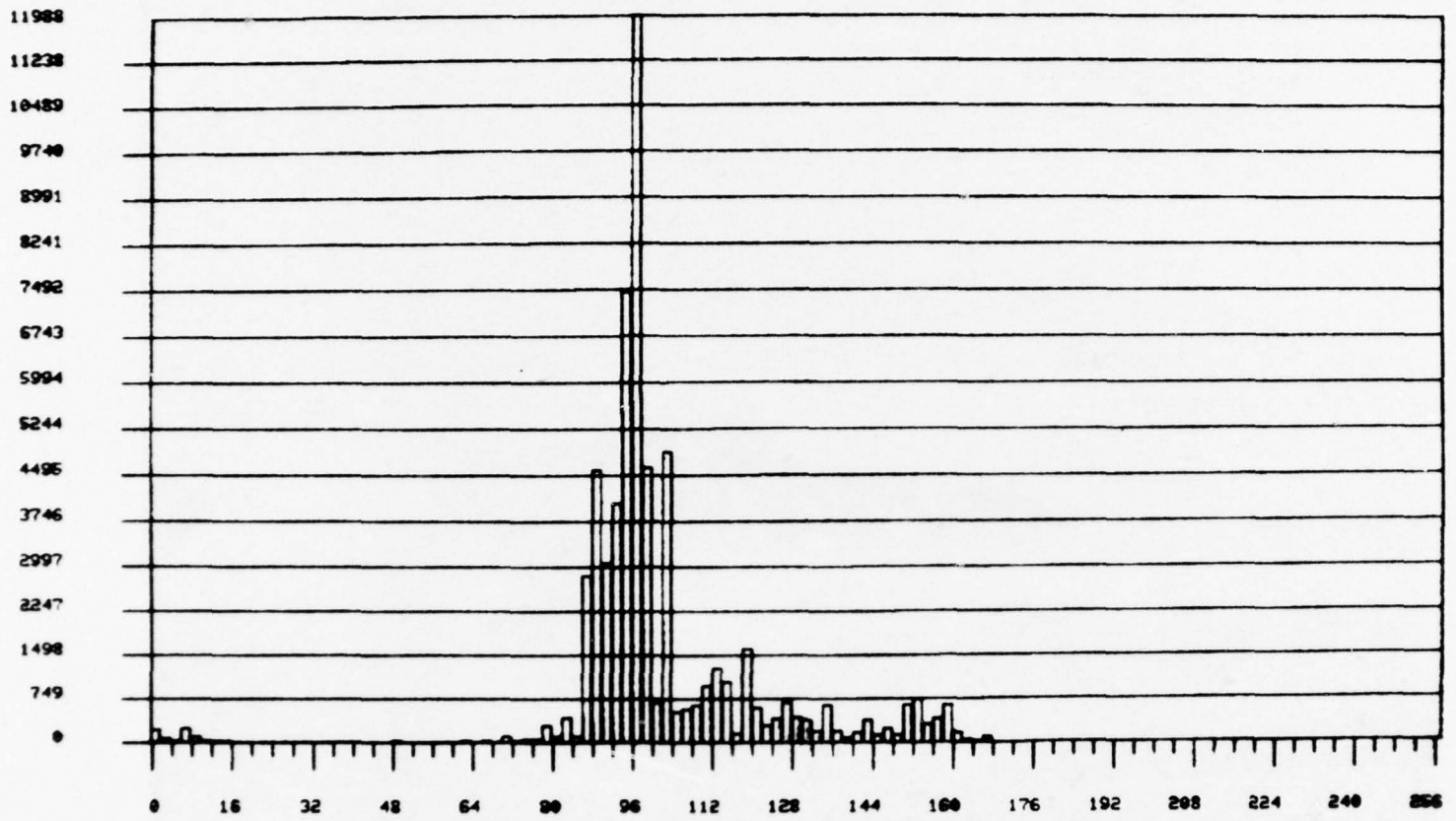


Figure 9.10b. Histogram of Missile 3, View 2

Figure 9.11a. Grey Scale Display of Plane 1, View 1

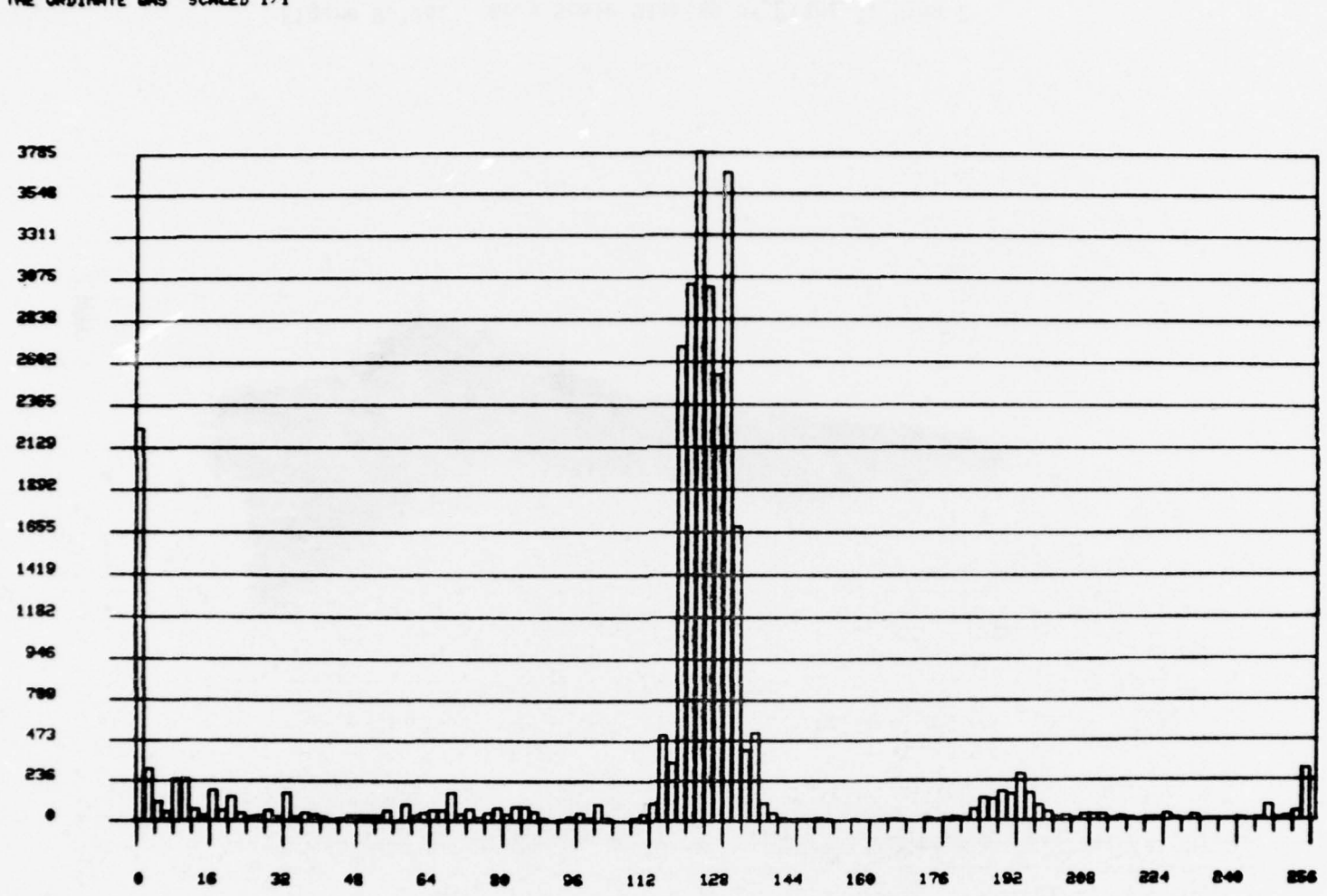


Figure 9.11b. Histogram of Plane 1, View 1

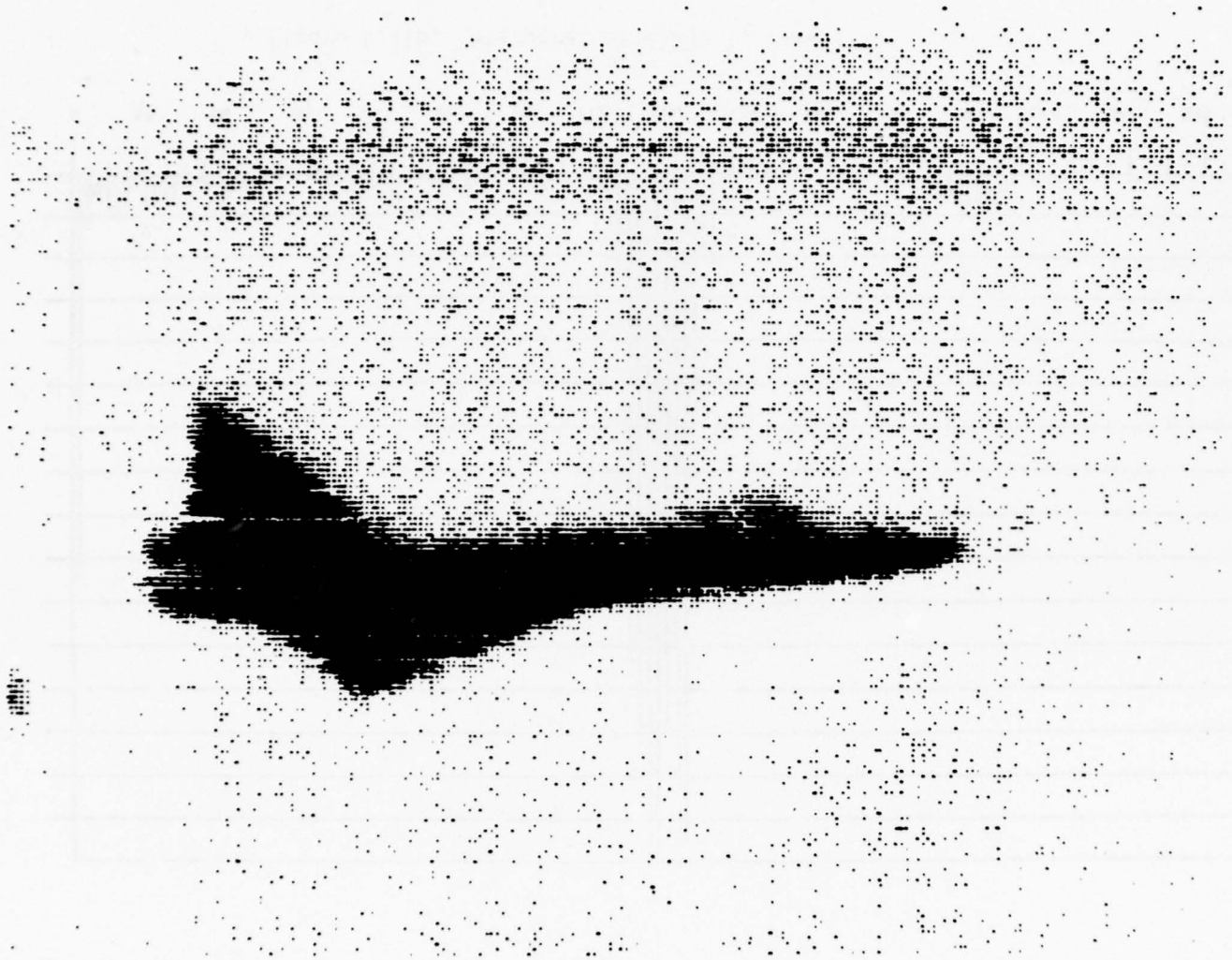


Figure 9.12a. Grey Scale Display of Plane 1, View 2

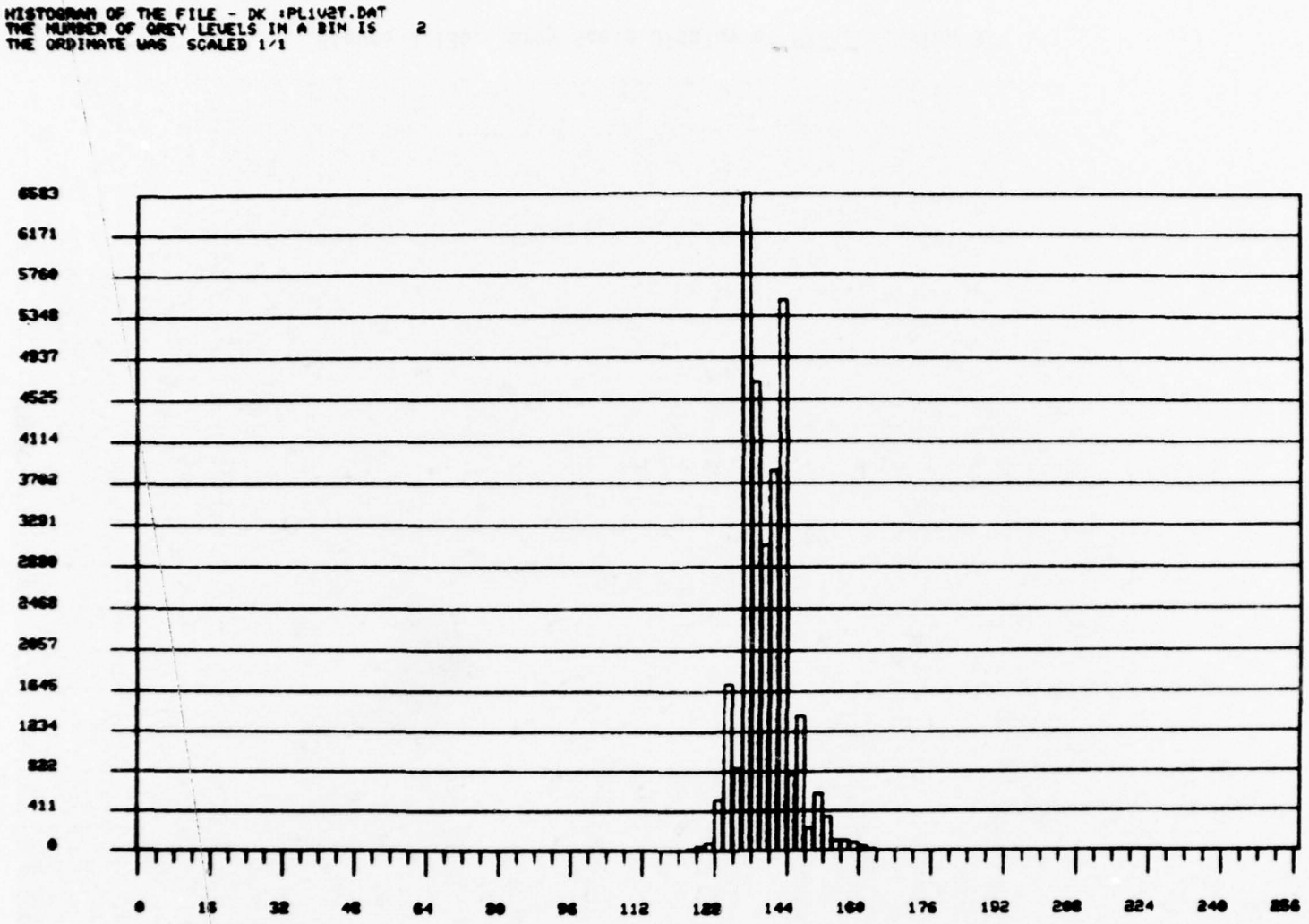


Figure 9.12b. Histogram of Plane 1, View 2

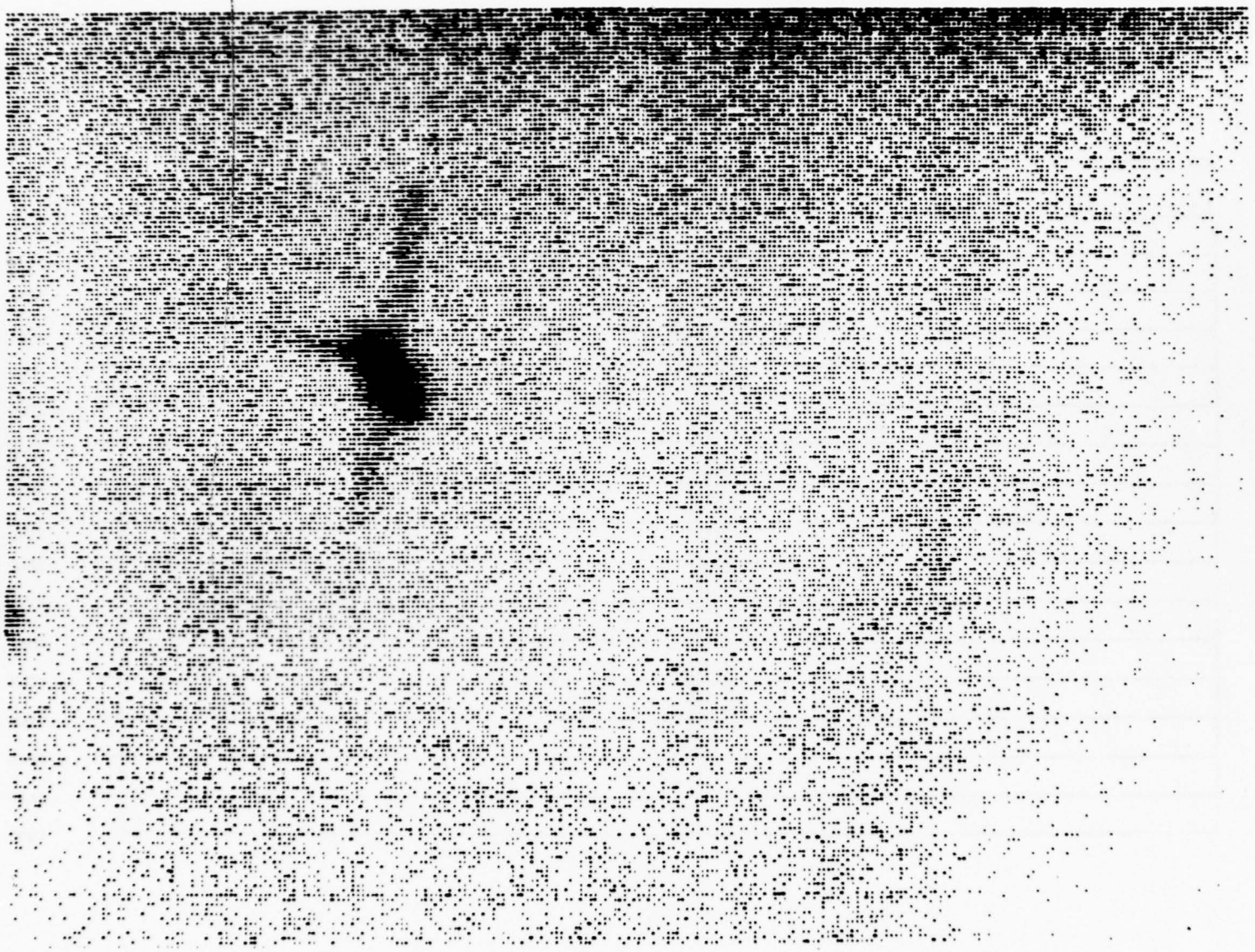


Figure 9.13a. Grey Scale Display of Plane 1. View 3

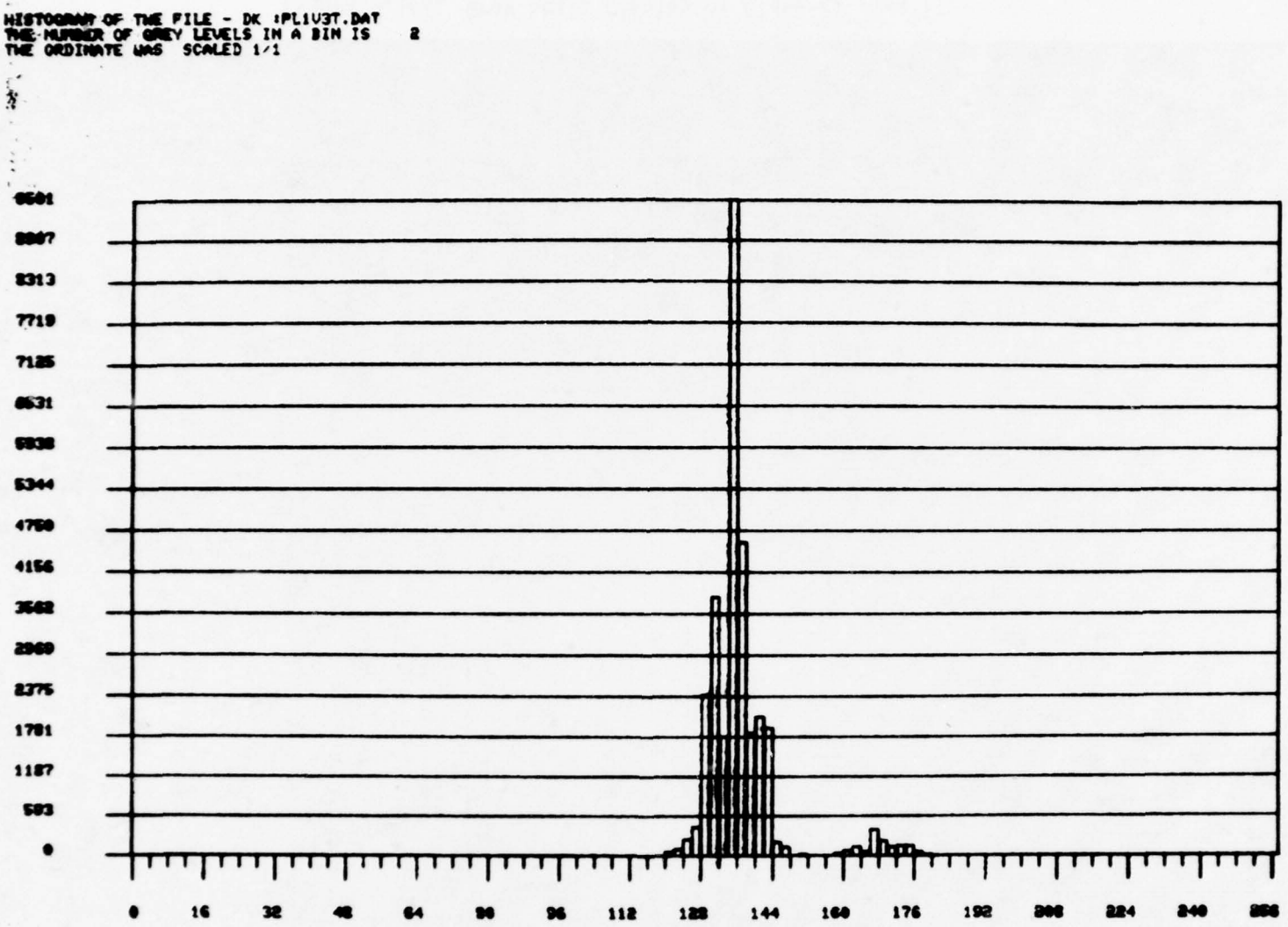


Figure 9.13b. Histogram of Plane 1, View 3

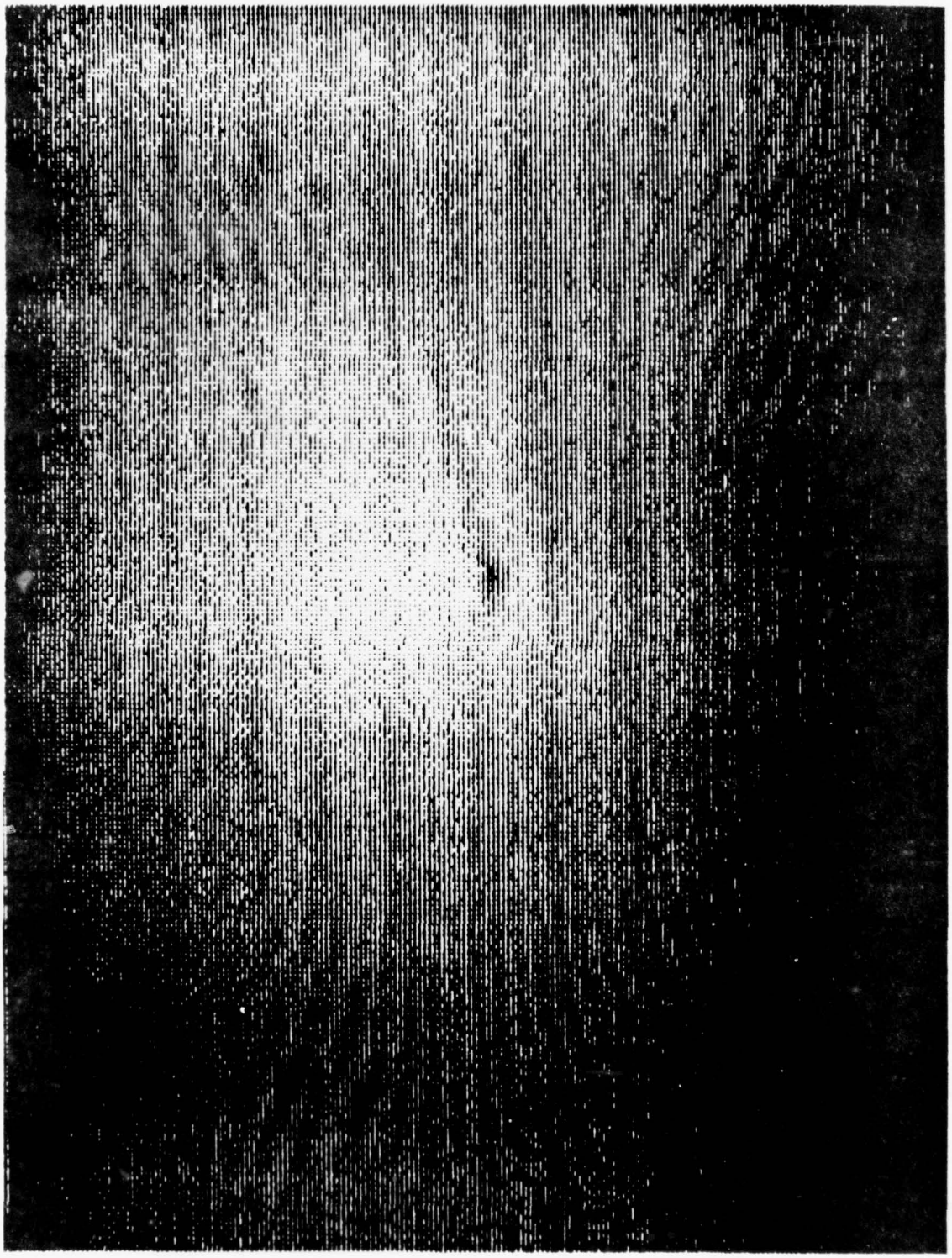


Figure 9.14a. Grey Scale Display of Plane 2, View 1

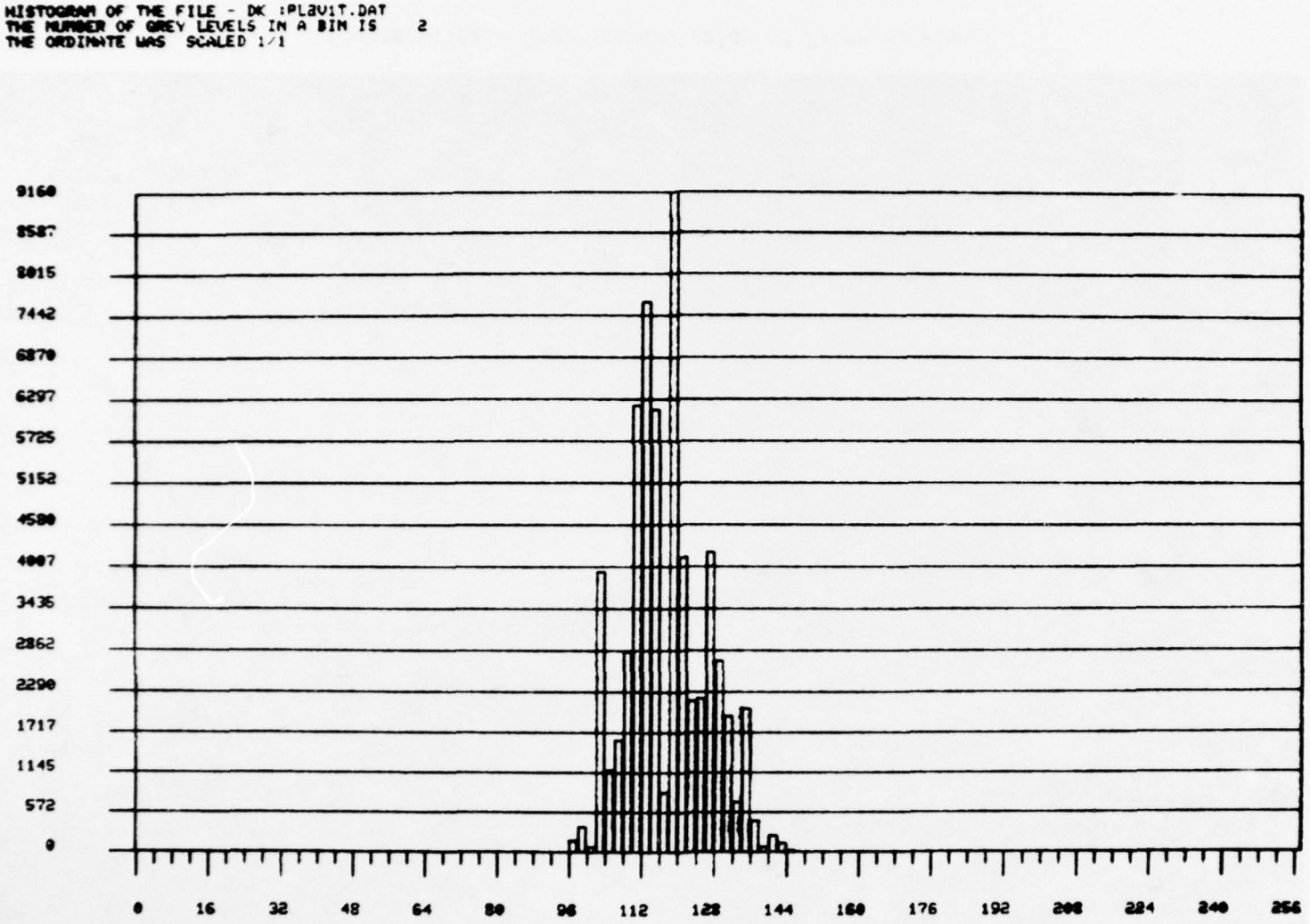


Figure 9.14b. Histogram of Plane 2, View 1

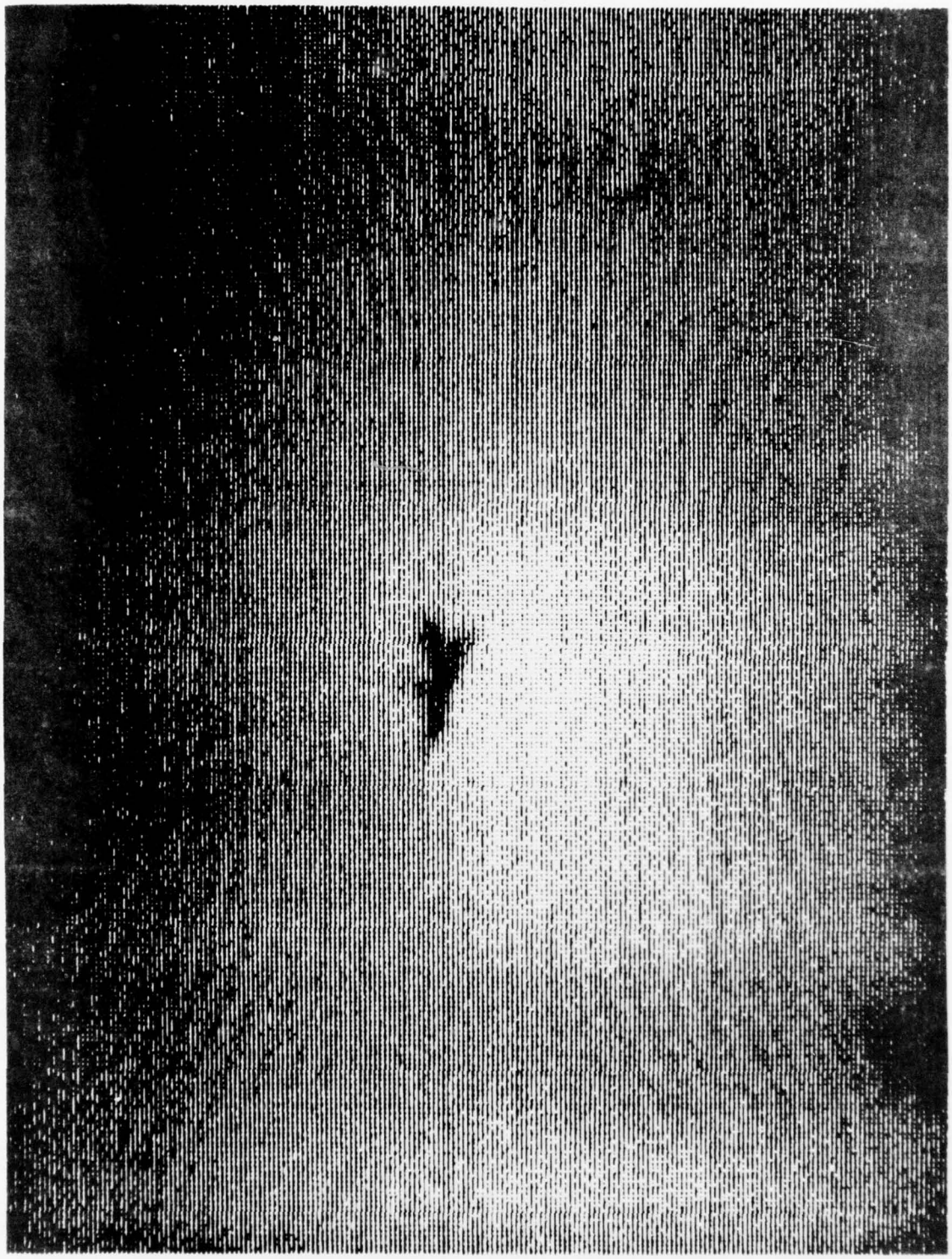


Figure 9.15a. Grey Scale Display of Plane 2, View 2

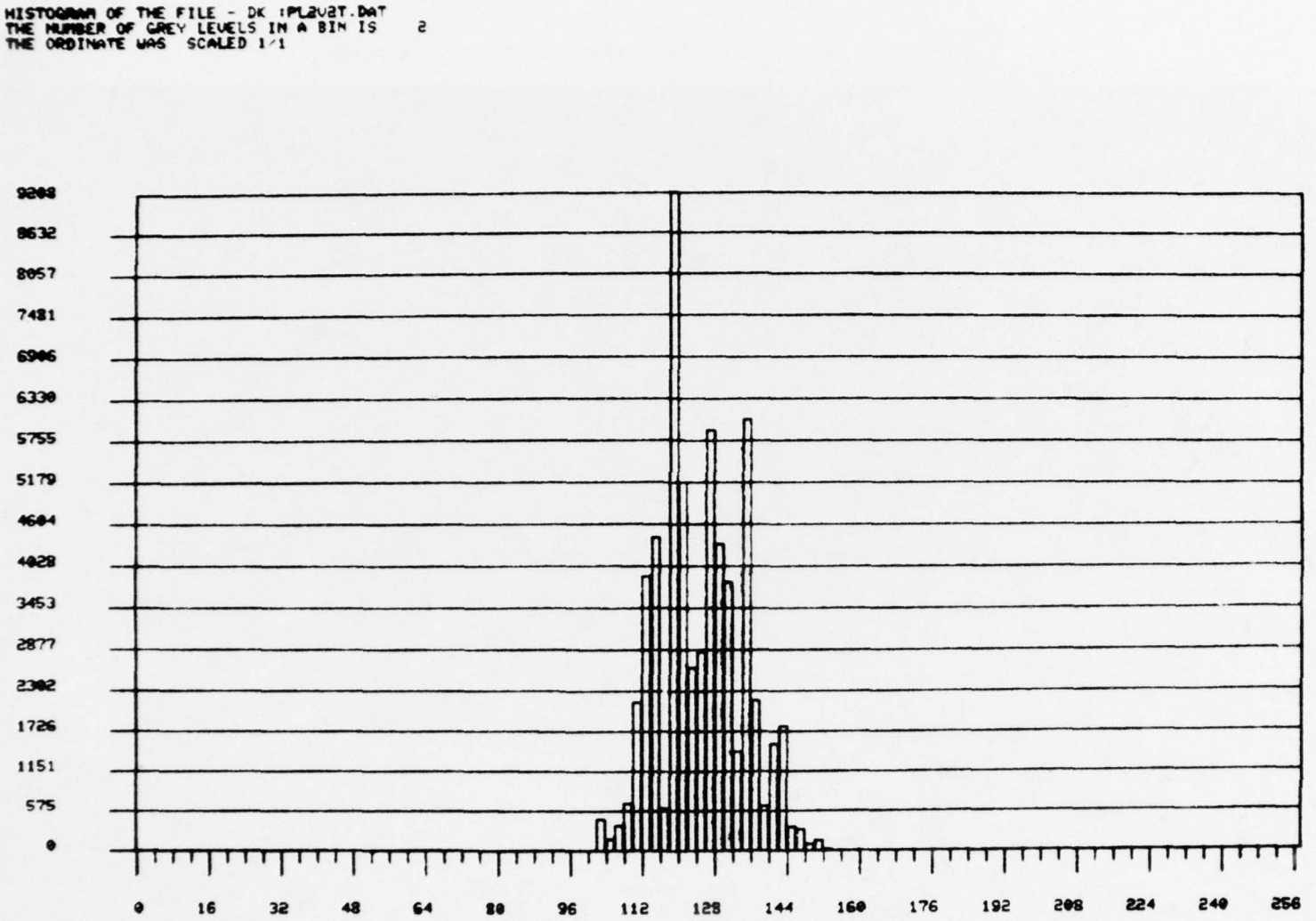


Figure 9.15b. Histogram of Plane 2, View 2

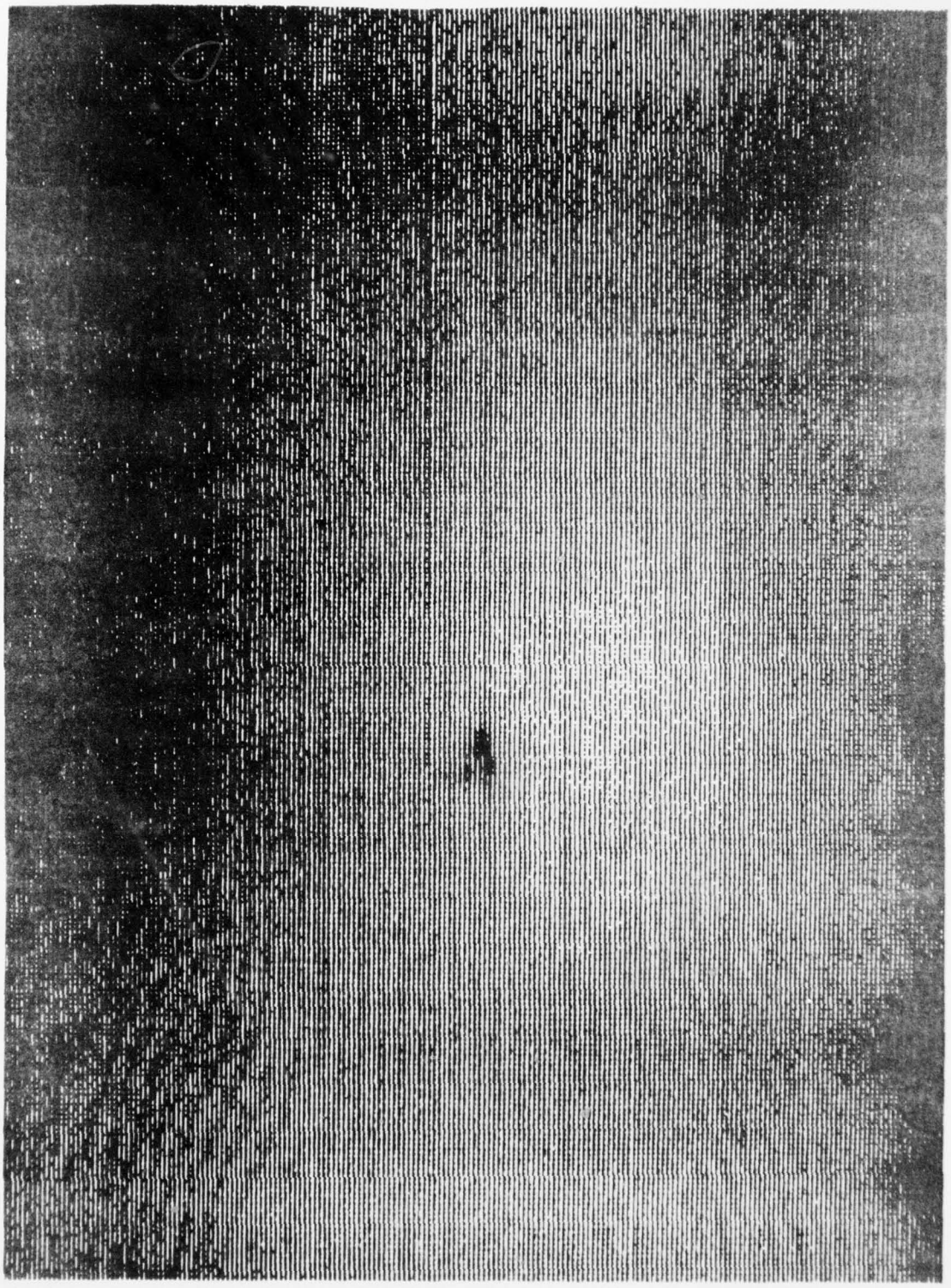


Figure 4.16a. Grey Scale Display of Plane 2, View 3

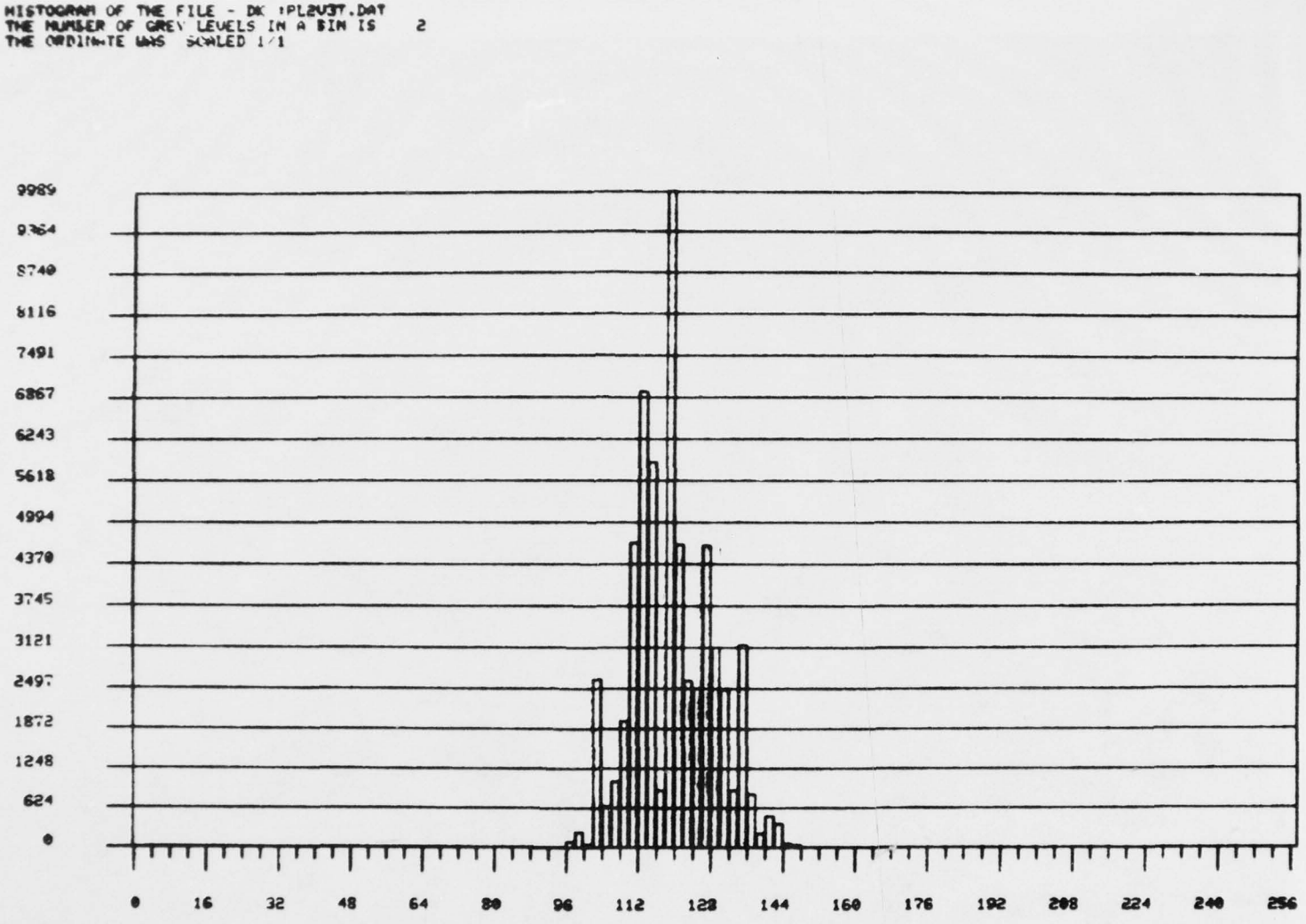


Figure 9.16b. Histogram of Plane 2, View 3

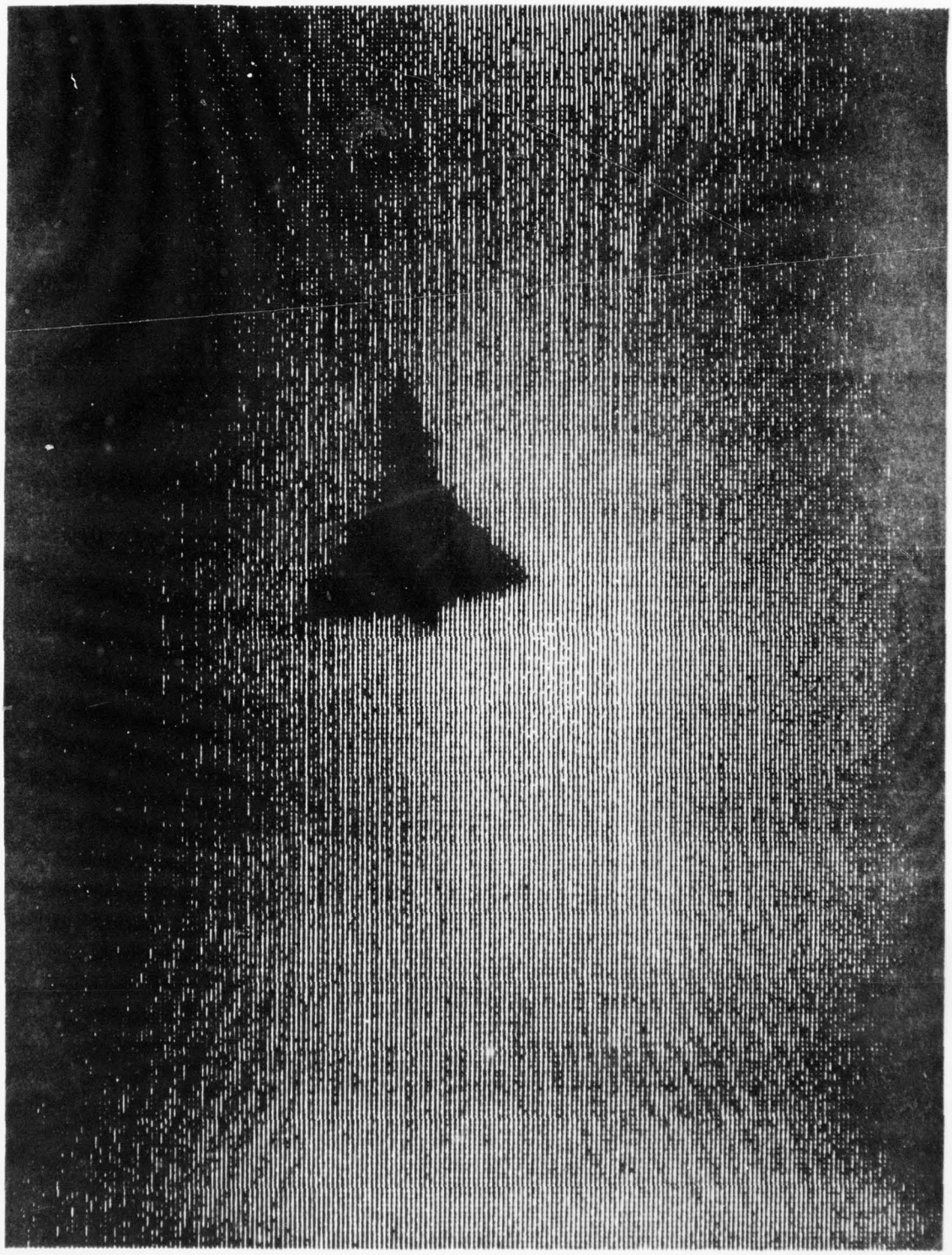


Figure 9.17a. Grey Scale Display of Plane 2, View 4

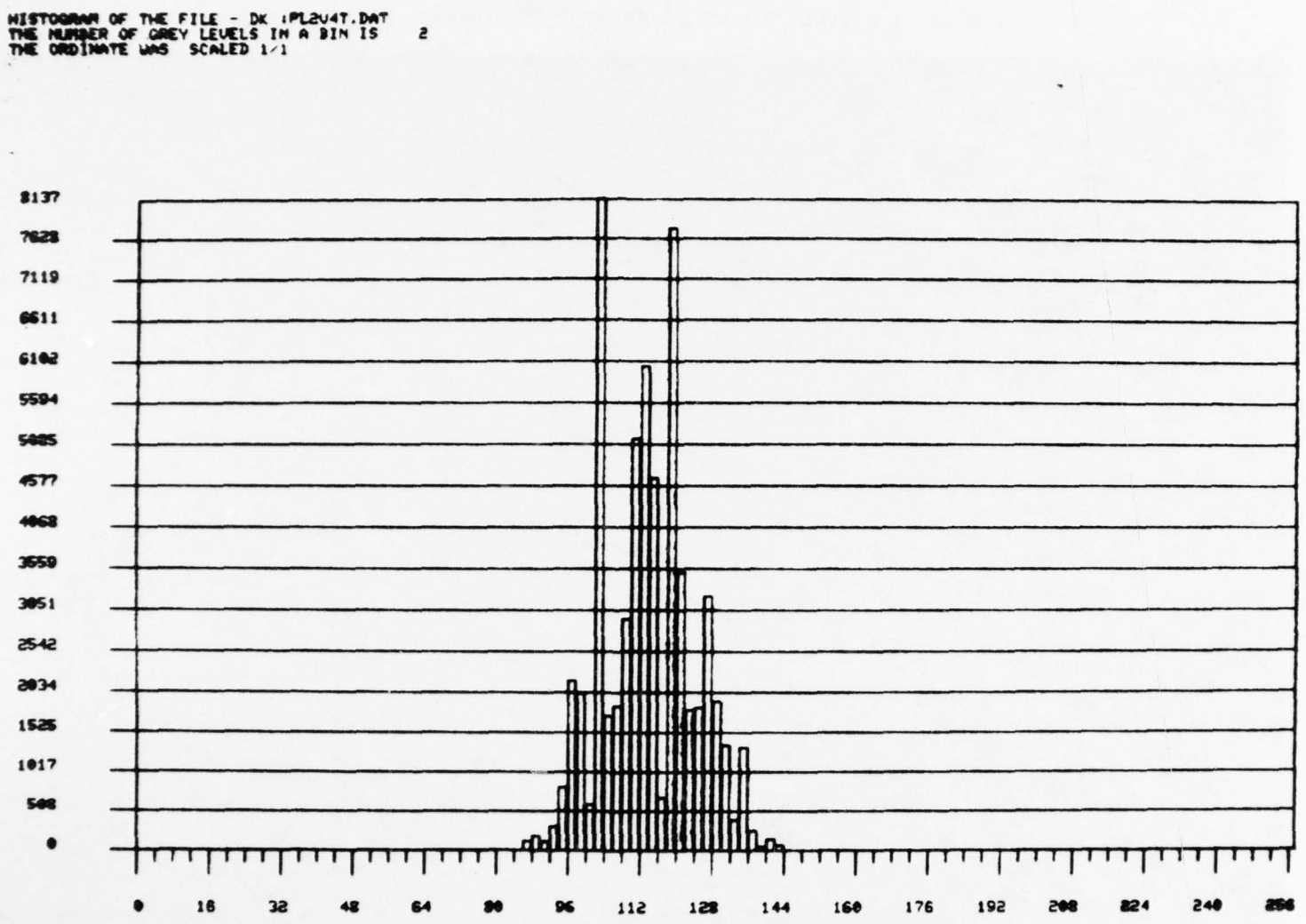


Figure 9.17b. Histogram of Plane 2, View 4

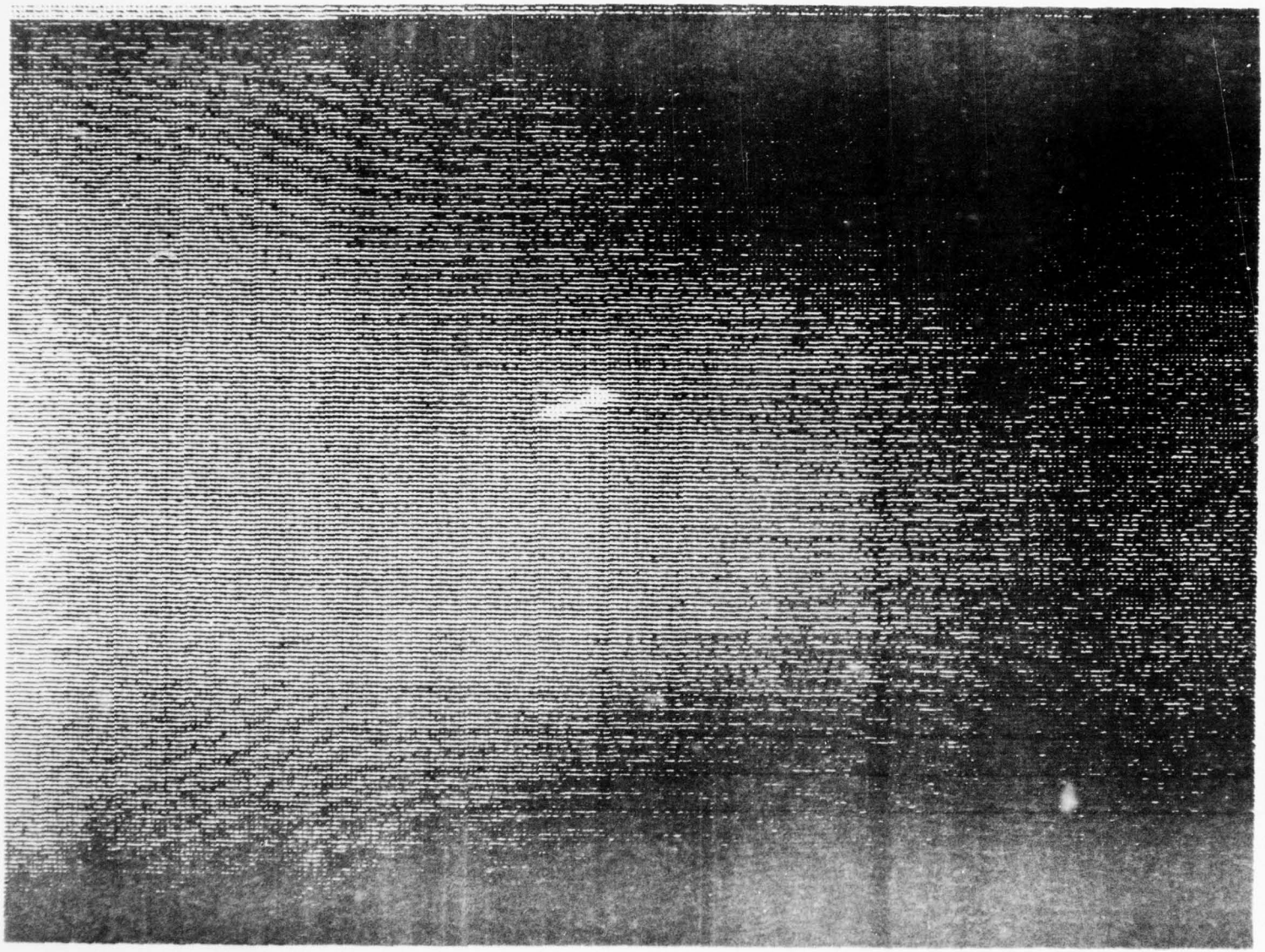


Figure 9.18a. Grey Scale Display of Cruise Missile 1, View 1

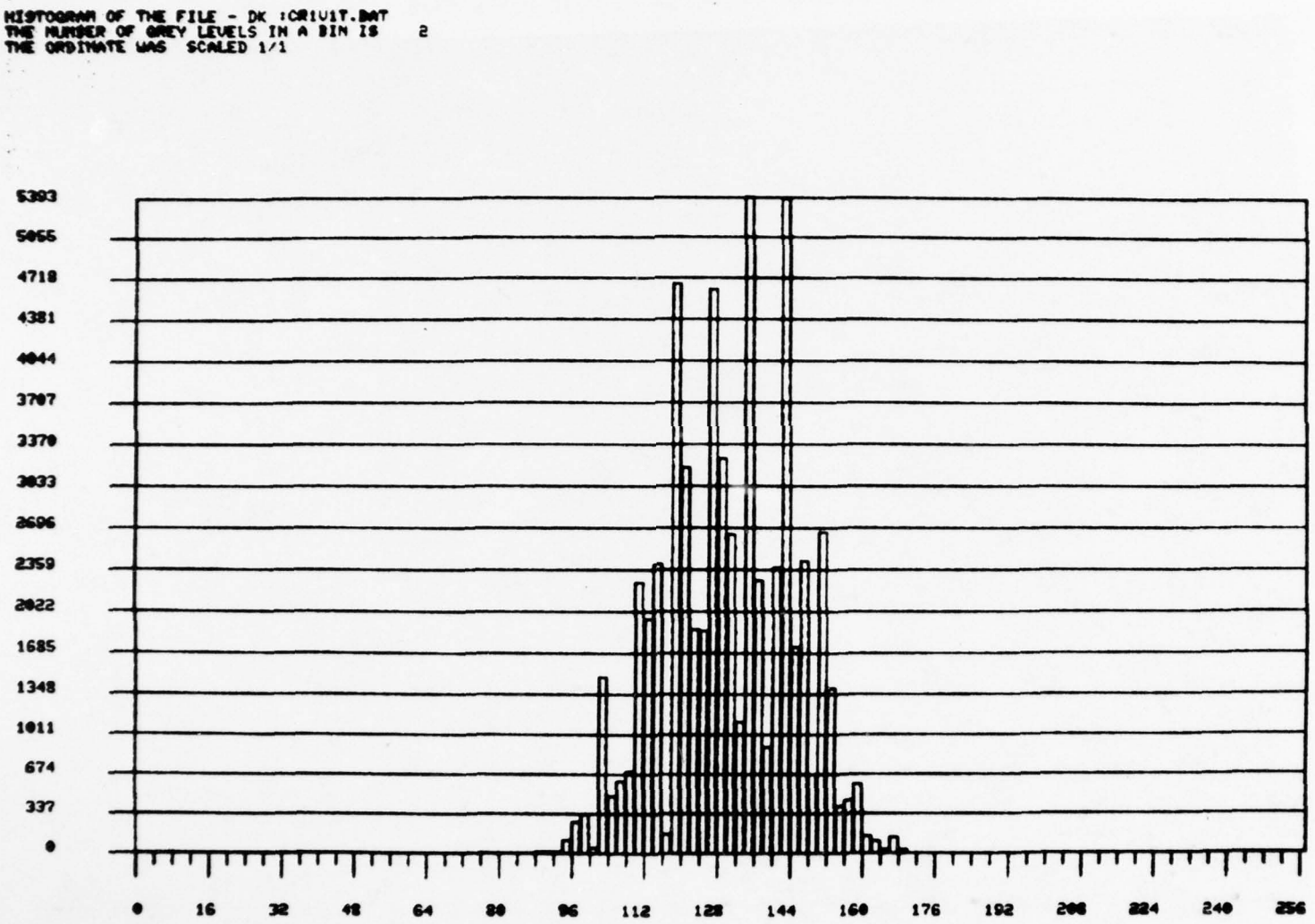


Figure 9.18b. Histogram of Cruise Missile 1, View 1

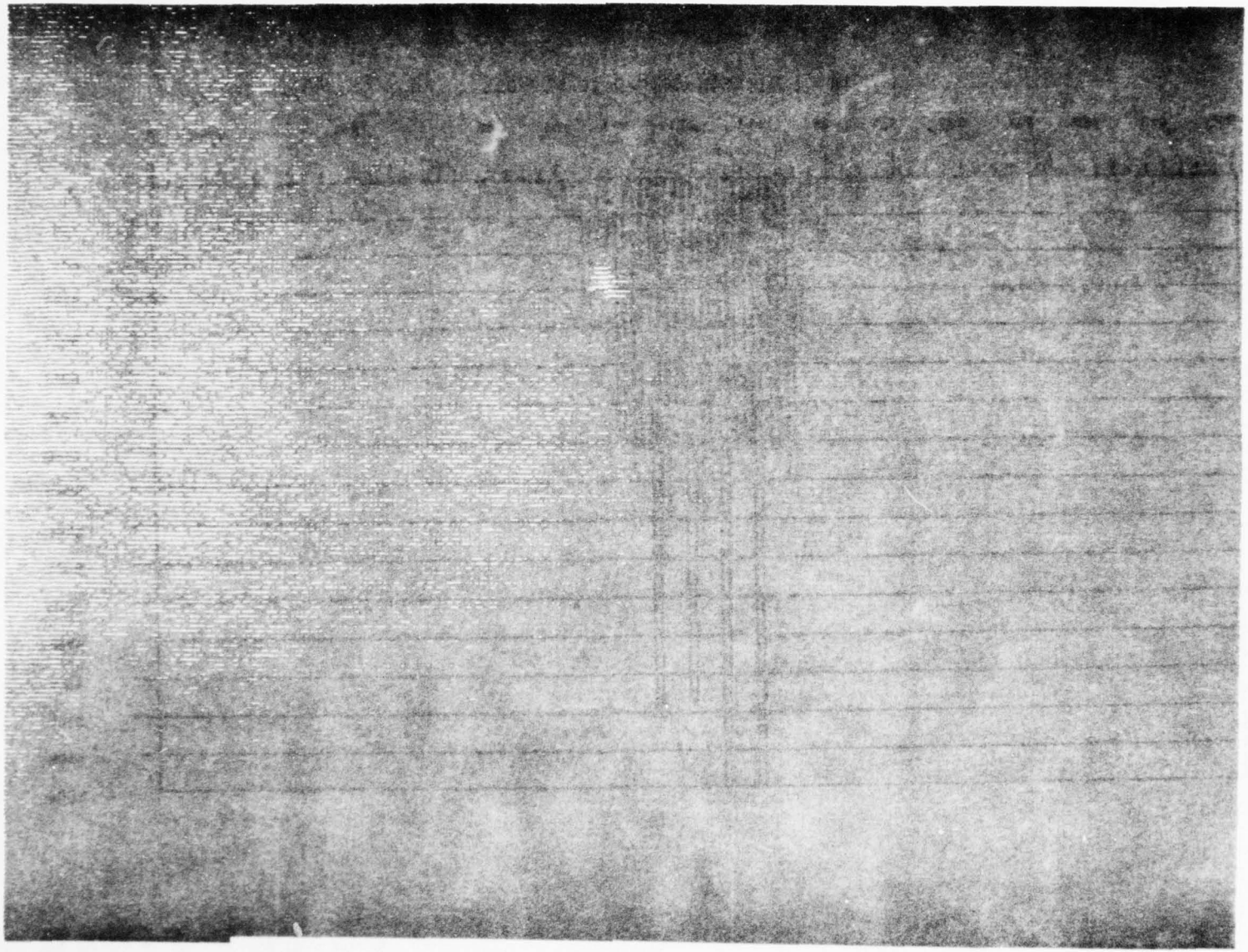


Figure 9.19a. Grey Scale Display of Cruise Missile 1, View 2

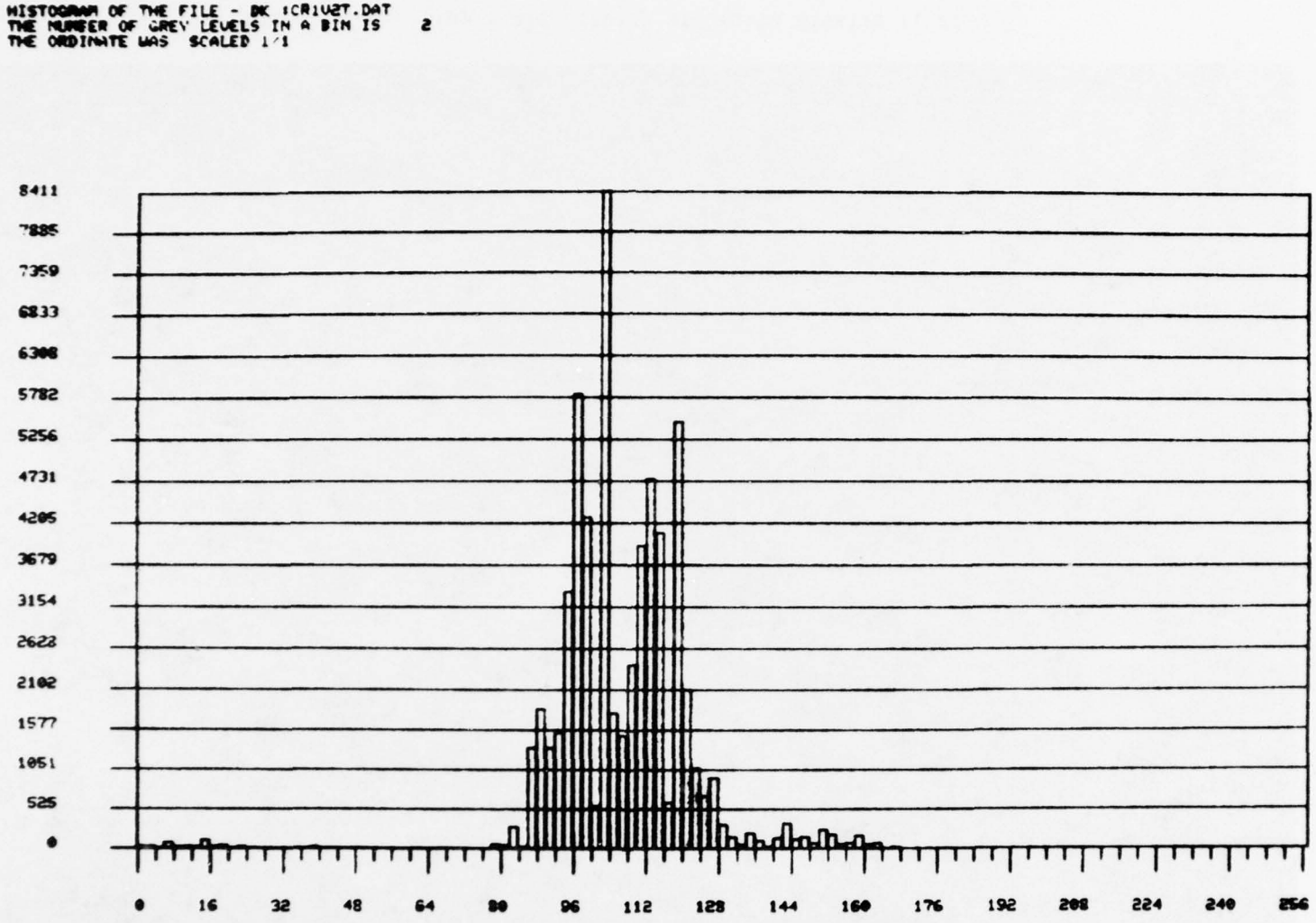


Figure 9.19b. Histogram of Cruise Missile 1, View 2

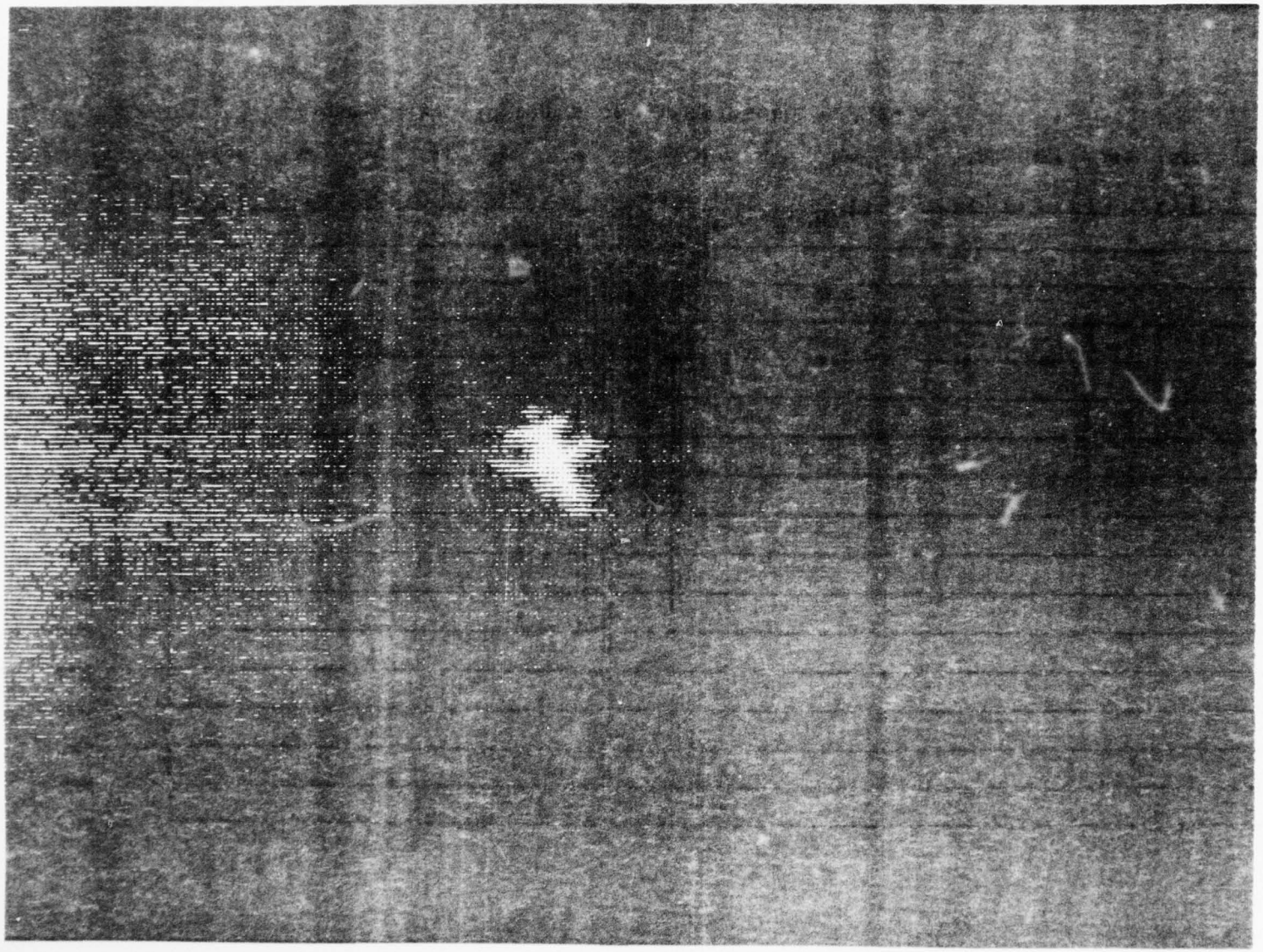


Figure 9.20a. Grey Scale Display of Cruise Missile 1, View 3

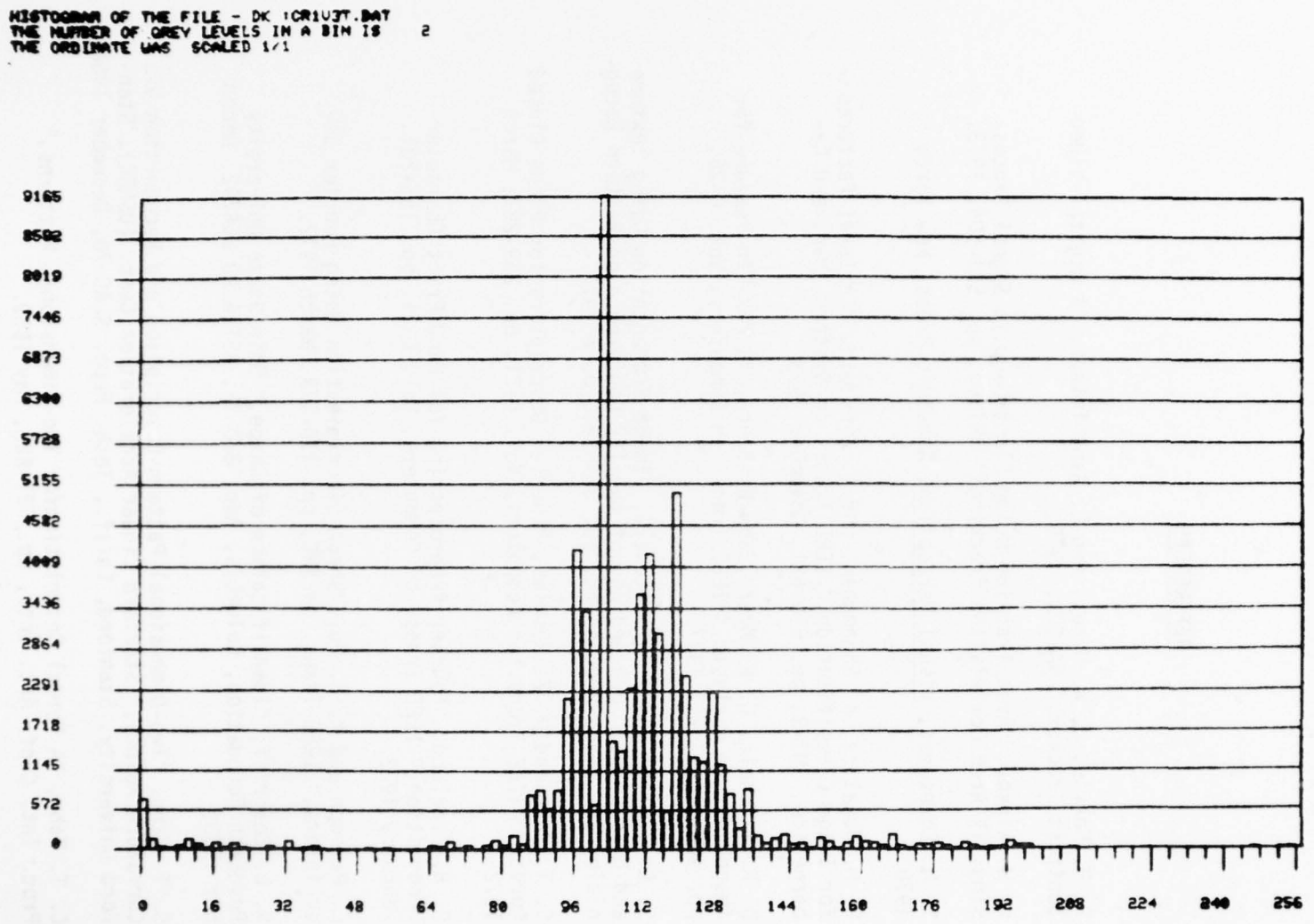


Figure 9.20b. Histogram of Cruise Missile 1, View 3

REFERENCES

1. R. H. Norris, J. W. Hines, and E. Ashenfelder, "A Digital Video Contrast Tracker," NAECON, 1975.

2. C. W. Winsor, "Area Tracking Using Electro-Optic Signal Processing; TV Area Correlation Tracking," SWIECO, pp. 501-504, 1973.

3.,7. T. N. Cornsweet, Visual Perception, Academic Press, New York, 1970.

4. R. M. Hanalick, K. Shanmugan, and I. Dinstein, "Textural Features for Image Classification," IEEE Trans. on Systems, Man, and Cybernetics, SMC-3, pp. 610-621, November 1973.

5. O. R. Mitchell, C. R. Myers, and W. Boyne, "A Max-Min Measure for Image Texture Analysis," IEEE Trans. on Computers, Vol. C-26, pp. 408-414, April 1977.

6. S. G. Carlton and O. R. Mitchell, "Image Segmentation Using Texture and Gray Level," Proceedings of the IEE Conference on Pattern Recognition and Image Processing, pp. 387-391, June 1977.

8.,12. C. T. Zahn and R. Z. Roskies, "Fourier Descriptors for Plane Closed Curves," IEEE Trans. on Computers, Vol. C-21, pp. 269-281, March 1972.

9. G. H. Granlund, "Fourier Preprocessing for Hand Print Character Recognition," IEEE Trans. on Computers, Vol. C-21, pp. 195-201, February 1972.

10. E. Persoon and K. S. Fu, "Shape Discrimination Using Fourier Descriptors," IEEE Trans. on SMC, pp. 170-179, March 1977.

11. R. L. Cosgriff, "Identification of Shape," Ohio State University Research Foundation, Columbia, Rep. 820-11, ASTIA AD 254792, December 1960.

13. C. T. Zahn, "Two-Dimensional Pattern Description and Recognition Via Curvature Points," Stanford Linear Acceleration Cent. (USAEC), Stanford University, Stanford, Calif., Tech. Repo. SLAC-70, December 1966.

14. C. T. Zahn, "A Formal Descriptionfor Two-Dimensional Patterns," Proc. Int. Conf. Art. Int., pp. 621-628, May 1969.

REFERENCES (Cont'd)

15. R. S. Leedley, Use of Computers in Biology and Medicine, New York: McGraw-Hill, p. 340, 1965.

16. U. Ramer, "An Iterative Procedure for the Polygonal Approximation of Plane Curves," Comput. Graph. Imag. Proc. 1, pp. 244-256, 1972.